MASTER THESIS

TITLE: PROPOSAL OF REMOTE VIRTUAL DESKTOPS ARCHITECTURE WORKING WITH THIN CLIENT DEVICES

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Josep Pujal Curià

DIRECTOR: Toni Oller Arcas

DATE: July 16th 2008

Title: Proposal of remote virtual desktops architecture working with thin client devices

Author: Josep Pujal Curià

Director: Toni Oller Arcas

Date: July, 21st 2008

Overview

This document contains an implementation proposal of remote virtual desktop accessed by thin clients.

A thin client device is a computer with low hardware resources; his embedded operating system that works as an interface to a remote desktop.

On the other hand, the use of virtual technologies permits to create and administrate dynamically remote desktops. From physically distributed offices, users can connect to remote desktops as locally. So there is no need to send the IT crowd to the foreign offices and corrupted thin client devices can be changed sending new ones by express mail.

This project faces the functional requirements and it proposes a scalable and optimum system architecture providing remote virtual desktops.

INDEX

INTRODUCTION

Nowadays most of the enterprises have all workers in front of a computer, those enterprises used to have regional offices physically distributed over the territory. There are also other cases, like education or goverment institution far from cities.

IT crowd travelling expenses are very important due to the hardware or software problems that happen on those locations, this project suggests a change of some habits relative to workplaces with clear objectives: cost reduction and better productivity.

It is decided to give the minimum relevance to the computer workplace, so a "thin client" device is installed. Thin client device is a small computer with low hardware resources, note that it does not have hard disk, only a flash memory to store an embedded operating system; it can be connected to network, keyboard, mouse, VGA, speakers and a even microphone. Some models only consumes 20 W. So if a device fails, the regional office can receive a new one by express mail, installation is very easy and there is no need to configure it.

Thin client embedded operating system is only an interface that provides to the user a remote desktop session. This remote desktop is a virtual operating system that has been instantiated from a virtual machine "seed", this virtual machine can be started, resumed, paused etc. depending on the user requests.

User can work thinking that his desktop session is working locally, but everything is executed and stored remotely; the fact is that desktops can be accessed by back office IT crowd and it can be restored or reconfigured easily. New updates are deployed from the service desktop provider and it is not necessary to configure each workstation, users can let his session paused and it can resume it even at home with his computer or laptop, only standard Internet connection is needed.

Before the first two chapters, an introduction to the current IT model is explained on chapter 0, afterwards, those first two chapters explains the functional requirements and system specifications from a conceptual point of view. Until chapter 3 no concrete technologies are presented and system architecture and implemented parts are described. Chapter 4 is a summarized business case based on this project. Environment impact has a lot of relevance mainly for the power reduction cost of this project, after this fifth chapter, future proposals can be found at chapter six.

CHAPTER 0. THE CURRENT "IT" PARADIGM

This chapter wants to talk about the current IT paradigm followed by most of enterprises, small business, universities, educational centres etc.

IT responsible has the order to provide a computer to every workplace, so the easiest way to do it is to buy N computers for N workplaces/workers. Every computer has his own individual resources that most of the time is not fully used: CPU and Hard disk usage are hardly used on most of the offices computers. In addition, if a new application needs to be deployed or just a new printer must to be configured, new modifications will have to be done to every single computer. Maybe it could be interesting to try to find a new model.

This project wants to minimize the workstation importance; if operating system is not executed locally, the computer device could need fewer resources, being cheaper and with less power consumption. So computer could be only an interface to a remote operating system, user won't be noticed and he will work like in a standard computer. This feature has an important bandwith expense, but nowadays broadband networks are generalized everywhere.

A cluster of servers could execute remote operating systems, so resources are shared and fully used thanks to load balancing techniques; here is a good place to use virtualization technologies.

Remote operating systems upgrades and modifications could be done easily because they are stored next to the IT back office support, so there is not necessary to send a technical expert to the workplace. Users could also connect securely to its remote desktop from any computer from the Internet and work with its personal files and directories.

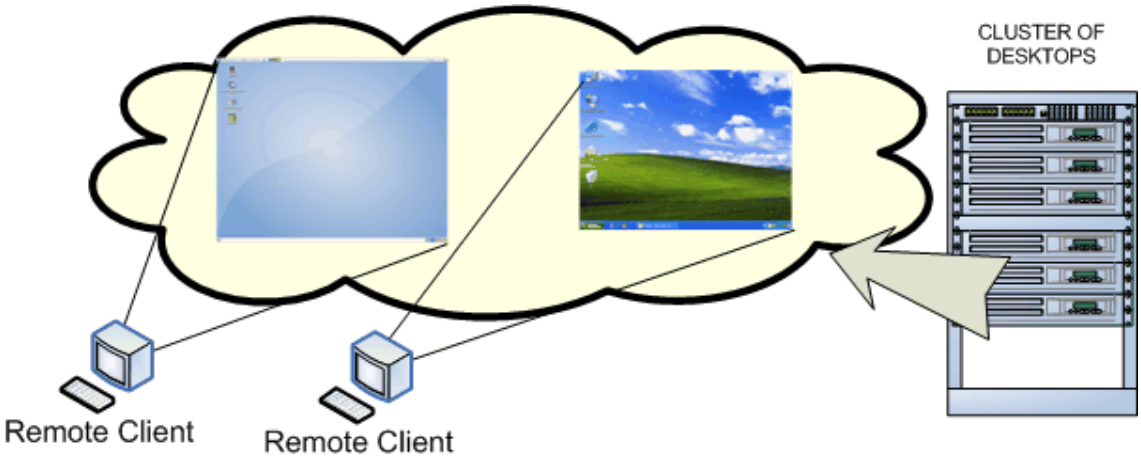

Fig. 0.1 The proposed model

CHAPTER 1. FUNCTIONAL REQUIREMENTS

This chapter examines the different challenges to be faced in order to achieve a successful start point. Those functional requirements do not mention any concrete technology and is only the first step to define the project intentions.

System functionalities are specified in three main blocks. The first block try to define the functionalities from the user's viewpoint, the user connect to an Authentication System and afterwards to a Platform, which will provide the work environment. The second one attempt to give a vision from the platform's administrator point of view; this section defines the gear's functionalities needed to give to the user the expected final service. The third block tackles the management and administration from Zero Client.

1.1. Zero client environment

1.1.1. Short description

The service lie in provide to a user a work environment from a minimum client device.

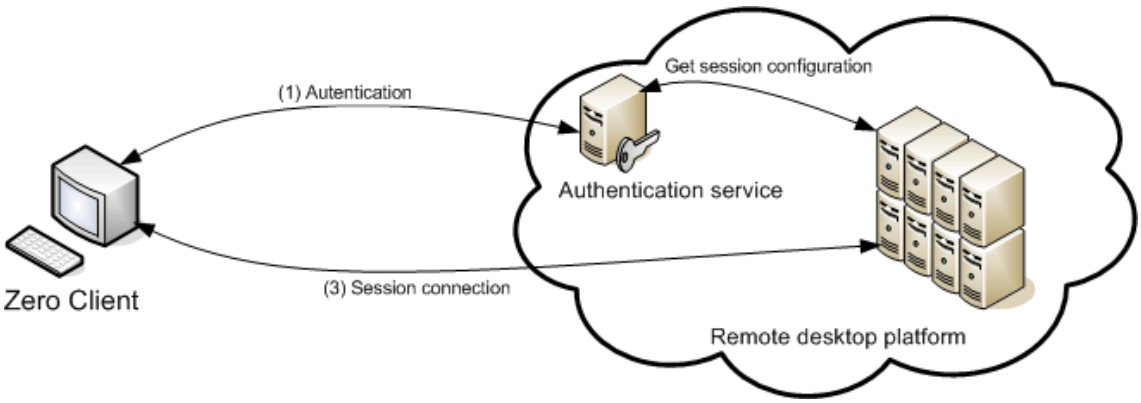

Fig. 1.1 Proposed service architecture

From the user client point of view, the system to design must provide a work environment as simple and functional as possible. This environment, based in office automation tools and software, must satisfy a set of expectations and provide added value in productivity.

It is worth mentioning in this subsection the hardware device characteristics found in the work environment, the client act quite simply like an interface in a remote environment. The device stands out from its simplicity giving the whole prominence from the service to the final platform. The connection is done in a clear, transparent way to the user and without any features loss; mainly it is expected to cause a sensation to the user that the operating system (OS) is

locally executing. That asymmetric service overshadowed the workstation, which in conventional systems, is the cause of the most office incidence.

One of the main characteristics from the client is that he is capable to connect himself to a final platform and then carry out an up-to-date of the mini-OS programmed by the administrator.

1.1.2. Primary actors

- Zero Client
- Authentication service
- Remote desktop platform

1.1.3. Actors specifications

- **Zero Client:** it authenticates and connects to the platform that will provide a remote desktop environment

- **Authentication service:** performs the access control to the desktop environment platform and gets the session information.

- **Remote desktop platform:** provides to a zero client device a remote desktop workspace, it will depend on the user profile defined by the authentication service

1.1.4. Pre-conditions

- Zero Client is found on a workplace with a keyboard, a mouse and a display

- Zero Client must have connectivity with the authentication service and the remote desktop platform

- Authentication is always necessary to connect to a remote desktop

- User wants to resume his last session

1.1.5. Post-conditions

- User wants to resume his last session if an error occurred (electric problem, connectivity failure etc.)

- Desktop environment can be restored if his integrity is compromised

1.1.6. Normal flow

From the user's point of view, these are the steps taken if a zero client wants to get a remote desktop session:

1. Zero client connects to the connection manager to get logged
2. Connection manager requests a remote session to the platform
3. Remote desktop platform returns the session information to the client
4. Zero client receives the session information parameters
5. If no successful authentication or any other error, an error message is shown
6. A zero client connects to a remote desktop using the session parameters; an assigned full working desktop appears into the zero client display.
7. The user disconnects the session

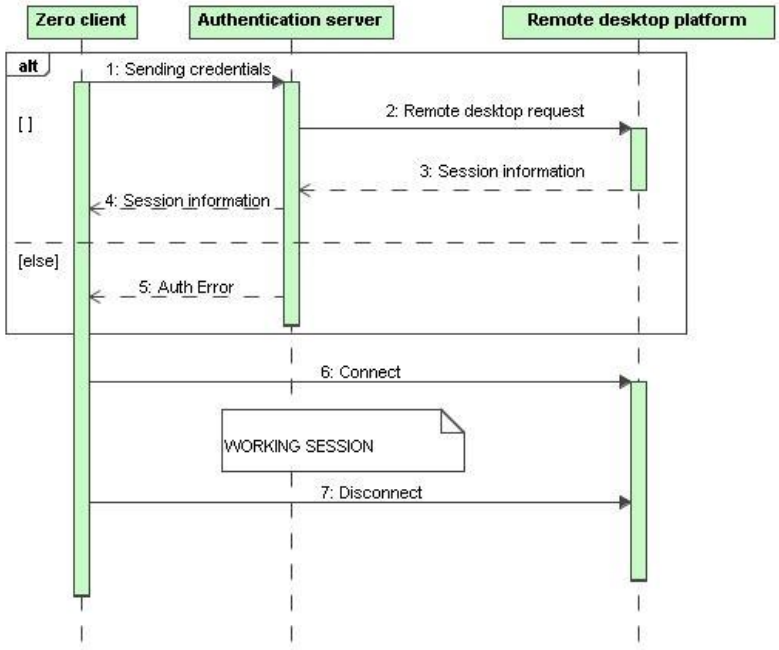

Fig. 1.2 A zero client interacts with the remote desktop platform

1.1.7. Cases of use

Mr Dwight Schrute works on a regional office of Dunder Mifflin Company, located in Scranton, Pennsylvania.
He arrives at the office, he sits down to the chair and finds a small device next to a display. He turns on the device and is prompted to enter his username and password. He wants to start working with his computer: check the email, web navigation and office suite tools usage; so after being logged, it appears a functional desktop session with its well configured applications.

1.2. Working environment platform

1.2.1. Short description

This platform offers a full working environment (desktop) when is requested by a Zero client device; this platform constitutes different services that interact each other to work

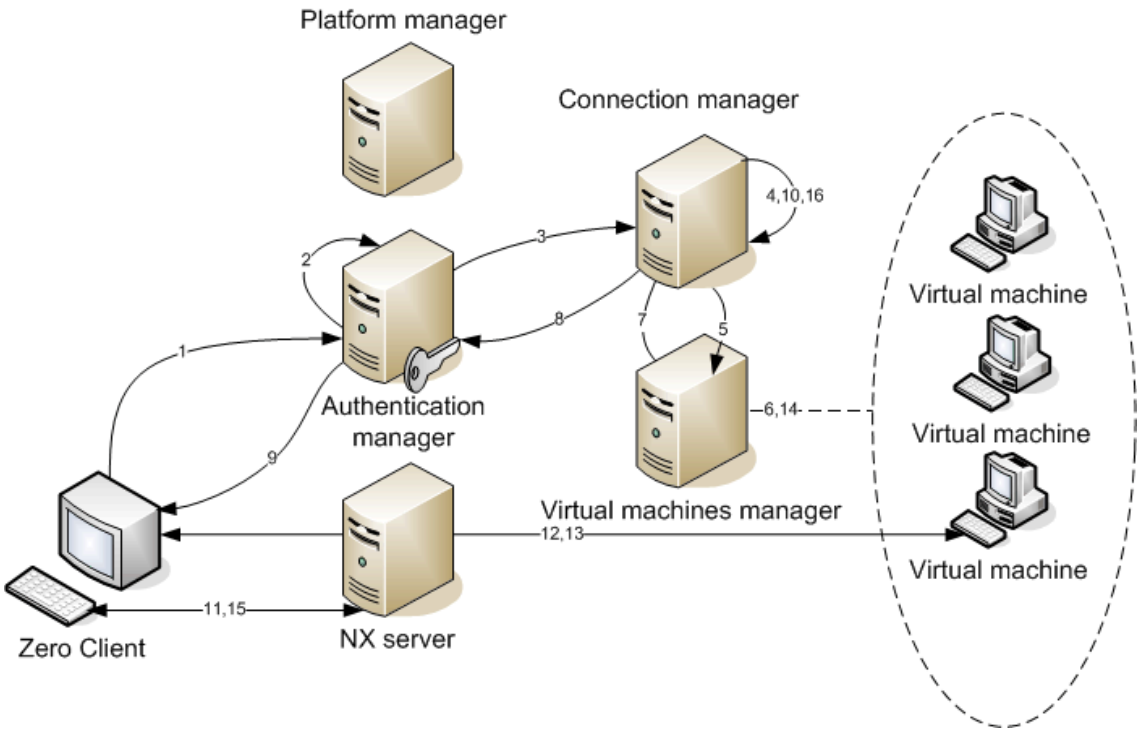

Fig. 1.3 Working environment platform

Working environment are originated from the invocation of virtual machines, they can be connected remotely by zero client devices. A connection manager is needed to perform control tasks to the virtual machines manager; it knows the resources state and the desktop sessions running. Virtual machine management service performs the translation between high level operations coming from the connection manager to the deployed virtual machine technology.

The platform is administrated with a web-based tool, it can manage: users, group users, user profiles, enterprises etc. and it can control: desktop sessions, platform resources and statistics. So the connection manager will work depending of the configured parameters by the platform manager.
NX server is the network gateway where zero clients are connected to.

1.2.2. Primary actors

- Connections manager
- Virtual machines manager
- Platform manager
- NX server
- Authentication manager

1.2.3. Actors specifications

- **Connections manager:** it monitors the virtual machine resources, performs load balancing tasks, informs about the sessions and it can administrate the virtual machines manager.

- **Virtual machines manager:** it can start, pause, resume, halt… virtual machines and informs to the connection manager about the new resources state.

- **Platform manager:** with a web-based interface it centralizes the whole service management and administrates: the access control layer, the running sessions and the service monitoring

- **NX server:** it is the network gateway where zero client connect to, it assures a secure channel between client and platform

- **Authentication manager:** it controls the access to the users. User's information is provided by the connection manager (type of session, permissions...)

1.2.4. Pre-conditions

- The working environment platform must have connectivity with zero clients.

- A user must to be registered (and associated to a group of users) if he wants to connect to the service. Depending on the user's information, he will be connected to the proper desktop session.

- Authentication manager connects a Zero client to the user’s session that previously has been logged.

- Connection manager admits only new sessions if resources are available

- Virtual machines manager is responsible of the desktop sessions and he sends/receives operations from connection manager and platform manager.

- Virtual machines manager has to be designed to work for different virtualization implementations. So it has to be defined an intermediate layer that will traduce the operations coming from platform manager to the respective virtual instance.

1.2.5. Post-conditions

- From the platform manager point of view, an administrator can monitor the resources state and it can manage the running desktop sessions (view, control, pause, restore...)

- Due to the heterogeneity and dynamic CPU load from the running sessions, connection manager could "migrate" sessions to other virtual servers inside the pool machines in order to maintain a desired quality of service.

1.2.6. Normal flow

When a user is authenticated (see section 1.1) from a Zero client device, these are the steps on a normal working flow:

1. User client connects to the authentication server to get logged
2. Authentication server consults the user profile
3. Authentication server requests to the connection manager a desktop session according to the user profile
4. Connection manager consults the available resources
5. Connection manager requests to the virtual machines manager a virtual machine according to the user profile
6. Virtual machines manager start/resume a virtual machine with a working desktop session
7. Virtual machines manager informs about the session information (type of session and IP address)
8. Connection manager informs to the authentication server
9. Client is being informed by the connection manager about session information
10. Connection manager updates the new resources state
11. Zero client connects to the NX server to establish a secured connection
12. Zero client connects to the virtual machine

(Desktop session usage)

13. User stops the session
14. Virtual machines manager stops the virtual machine

15. Nxserver disconnects the client
16. Connection manager releases the used resources; now those can be used by any other user.

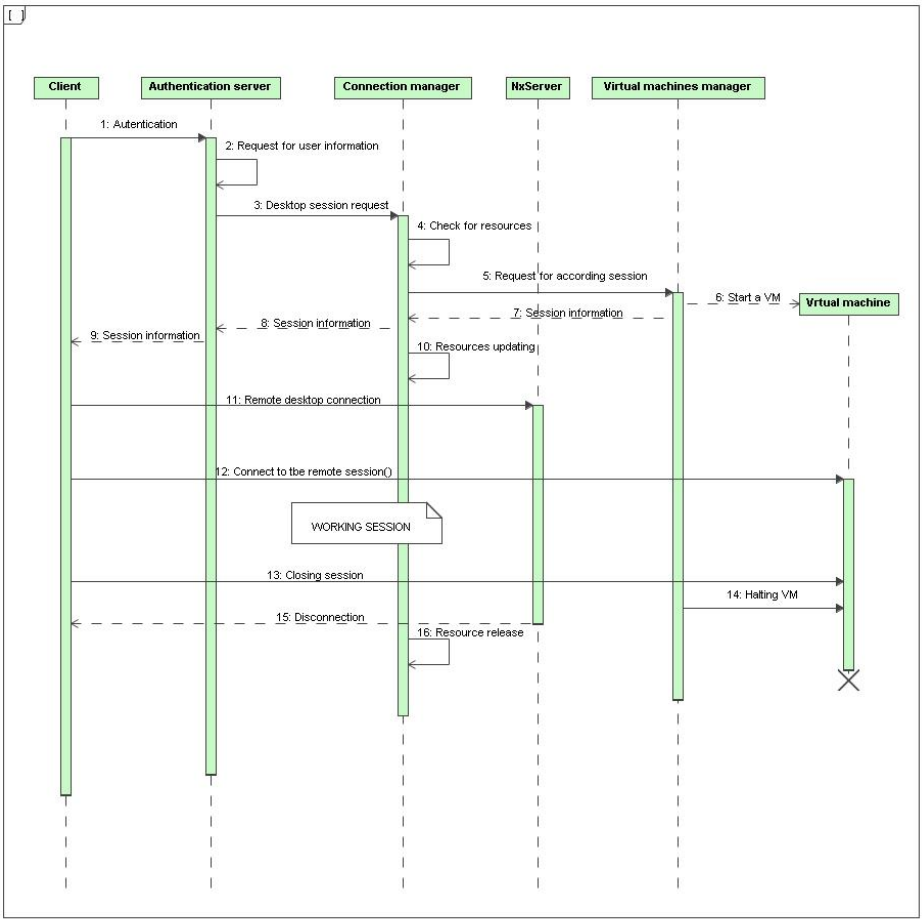

Fig. 1.4 Normal flow

1.2.7. Cases of uses

Michael Scott is the Regional Manager of Dunder Mifflin Company, located in Scranton, Pennsylvania. As the responsible, his objective is to receive a good IT service.

If a computer is down, he does not want to wait for a person to come and repair the device, he does not want to be worried about the local information stored on the machine.

So if computers are changed by zero clients, the device can be changed immediately and a new one can be received by express mail service. In addition, information and desktop environment is stored on the remote servers, so time to repair is very small.

Mr Scott's workers sometimes execute nasty programs and they are used to break the operating system, so with only a call to the IT front office, the working desktop can be restored.

1.3. Thin client embedded OS management

1.3.1. Short description

This chapter wants to describe the elements that will provide configuration and support to the zero clients situated into remote offices.

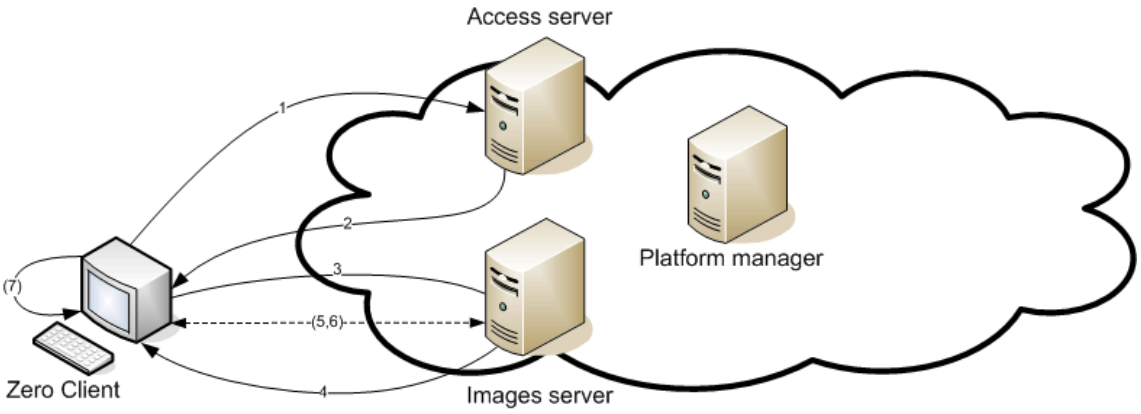

Fig 1.5 The Zero client platform manager

The main objective permits to the platform administrator to manage the remote user access control layer (access control server), performs operating system maintenance and deployment tasks to the zero client devices (images server); all of it via a web interface.

1.3.2. Primary actors

- Zero client
- Access control server
- Images server
- Platform manager

1.3.3. Actors specifications

- **Zero client:** it runs an embedded operating system and it receives network parameters from the access control server

- **Access control server:** sends network parameters to the zero clients according to the configuration set by the platform manager

- **Images server:** it deploys and updates the zero client's embedded operating system

- **Platform manager:** like on previous chapters, it offers a web interface to manage the access control and software images servers.

1.3.4. Pre-conditions

- The remote office network and service provider network must have level two connectivity.

- A zero client should be installed with no technical knowledge and giving service from the first day

- A zero client device must to be registered into the platform manager if it wants to get network parameters and being able to connect to a remote service

- New zero clients restored by factory must come with a initial embedded operating system; it is not efficient to download a entire software image from a remote office

1.3.5. Post-conditions

- An operating system image update must control the network bandwidth usage to ensure the quality of service of the other clients working on the same remote office. Incremental download technologies should be used.

- The embedded operating system is read only, if a user forces to write to the system partition, all the files will be restored after a device reset.

- A zero client device has two embedded operating systems, each one on a different partition. The first one contains the primary OS which is used by default, the second one is used to be updated. When the second one is updated, it is swapped with the primary OS and it will be used after the next reboot.

- Update procedure is executing while a user is working, those tasks must run transparent and with no service affection.

- No user data is written into the local disk, all the documents are remotely stored

1.3.6. Normal flow

This is the normal flow where a zero client interacts with the access and software image server:

1. Zero client requests for network parameters to the access server
2. Server access returns to the client device the network parameters, now they have full connectivity.
3. Zero client asks for a new embedded operating system version
4. Zero client compares with local and remote version
5. If there is a new one, it proceeds to download the incremental files
6. Downloading files
7. Boot loader modification swaps the primary partition with secondary and vice versa, it will be no changes after next reboot.
8. In case of no new updates, the embedded operating system is not modified

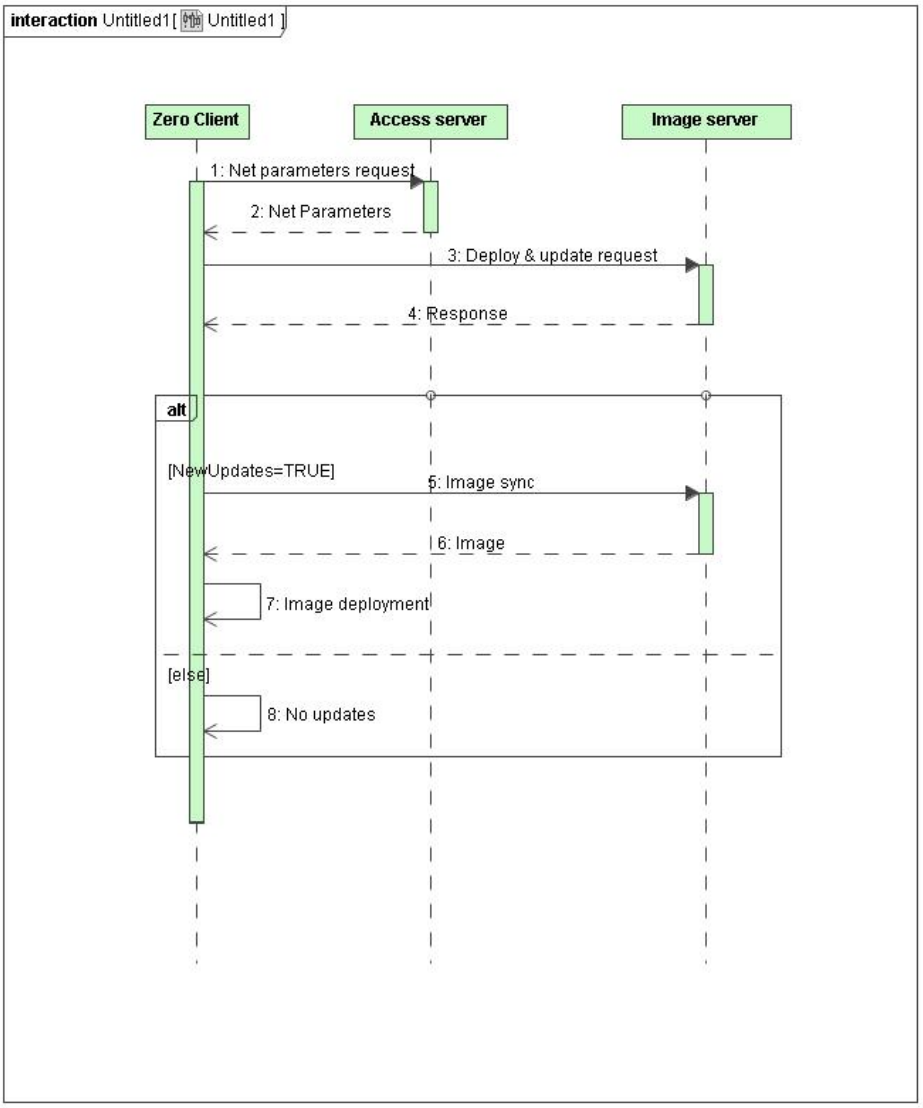

Fig 1.6 Interaction diagram

1.3.7. Cases of uses

Mr Schrute does not know that when he turns on his zero client device, it gets the network parameters and afterwards, during the working desktop session, the device updates the embedded operating system.

CHAPTER 2. SYSTEM SPECIFICATIONS

This chapter is based on the functional requirements defined on chapter 1. Diagrams are the quickest way to explain system specifications, UML [1.1] and RM-ODP [1.2] models are used.

2.1. System architecture

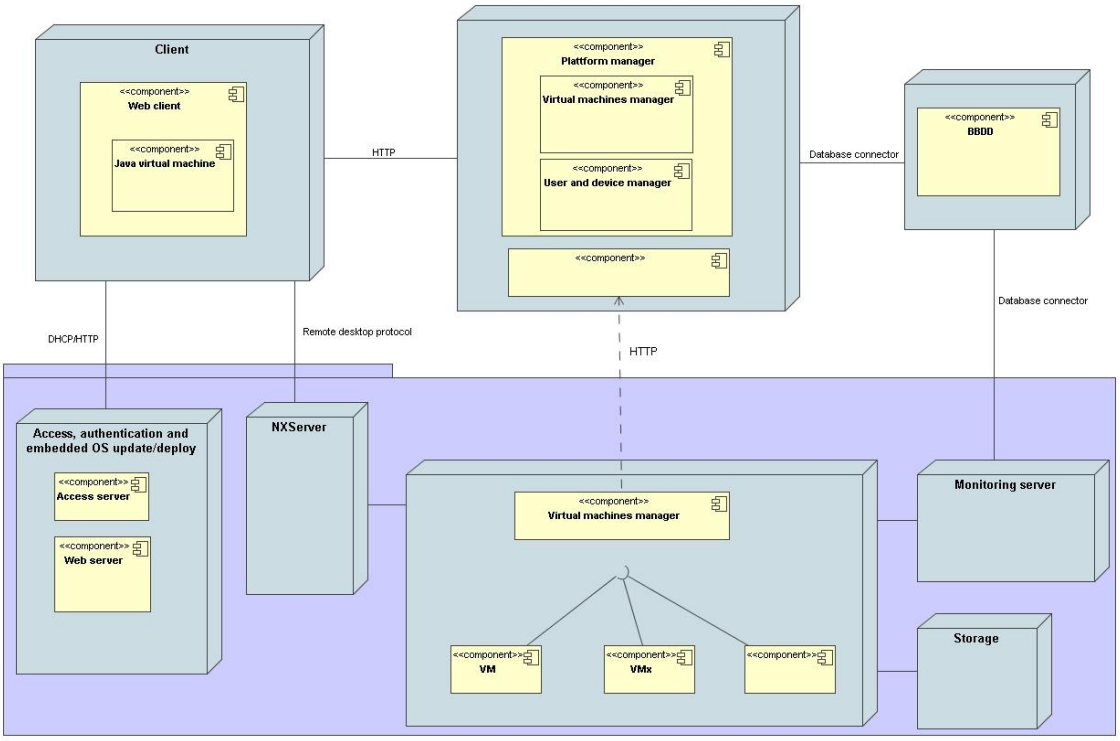

Fig 2.1 System architecture

System architecture definition described on this section wants to combine all the functional requirements described on section 1; no specific technologies are yet described, this will happen on section 3.

- **Client:** it has a web browser component with the capability to run java applets.

- **Platform management:** this node offers a web application that configures the global parameters and manages the virtual machines manager. It has a component known as the connection manager that manages client's requested sessions.

- **Access and authentication:** this node has a web and an access server that sends automatically the network parameters. Web server provides

authentication service and embedded operating system software deployment.

- **NX server:** it is the thin client gateway

- **Virtual machines manager:** this component is waiting for remote operations coming from the platform management; those operations calls to the appropriate application layer that for example could start virtual machines.

- **Database server:** this node stores the access control layer (ACL) information and system resources, the last one is used to balance desktop sessions.

- **Monitoring server:** collects the system state resources and it stores to the database server.

- **Main storage:** it stores virtual machines, configurations and user personal files and directories.

2.2. Use case realizations

This section shows the use case realizations. The first one show how a user interacts with the platform and it connects to a remote desktop, after been working with it, session is disconnected.

Next sub-section shows the use case realizations during the embedded operating system deploy and update procedure.

2.2.1. User and platform point of view

Figure 2.2 shows the element interaction during the starting session:

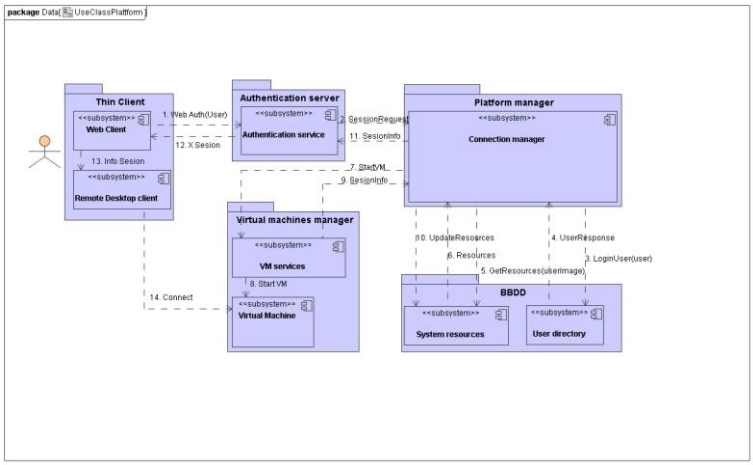

Fig. 2.2 Use case session start

Figure 2.3 shows the element interaction during the ending of a session:

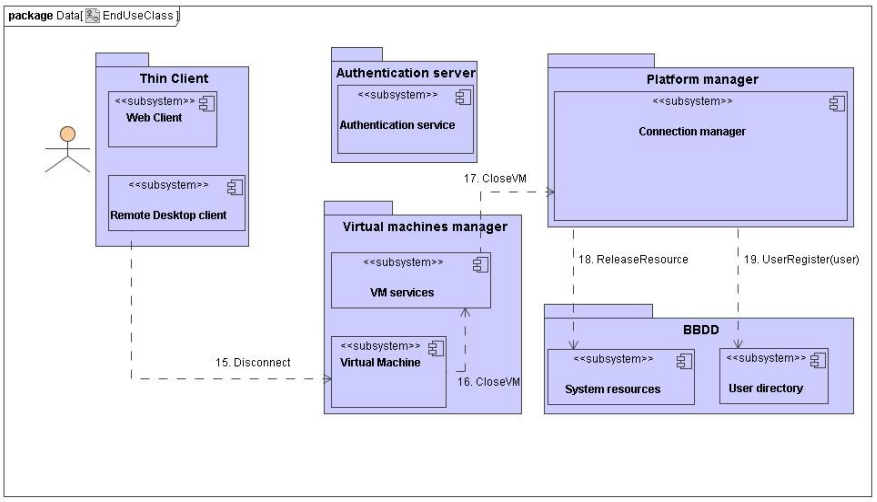

Fig. 2.3 Use case ending session

2.2.2. Thin client device point of view

Figure 2.4 shows the interaction between the thin client device, the access server and the update/deploy OS server.

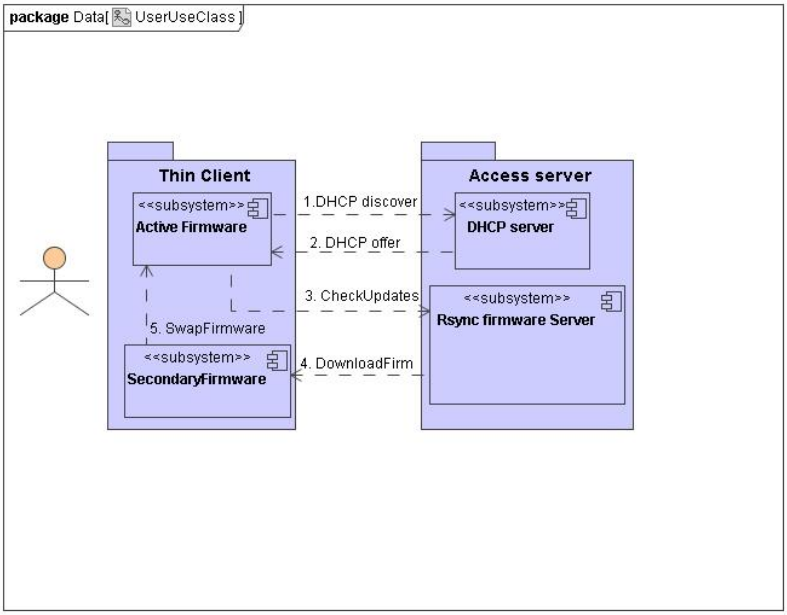

Fig. 2.4 Thin client use case

2.3. Class Diagram

In the Unified Modelling Language, next figure corresponds to the structure the system showing the system's classes, their attributes and their relationships between classes.

Group of users has multiple users and one enterprise has multiple of them. An enterprise has multiple groups of workstations and each group has multiple workstations. A desktop image is linked with each user group.

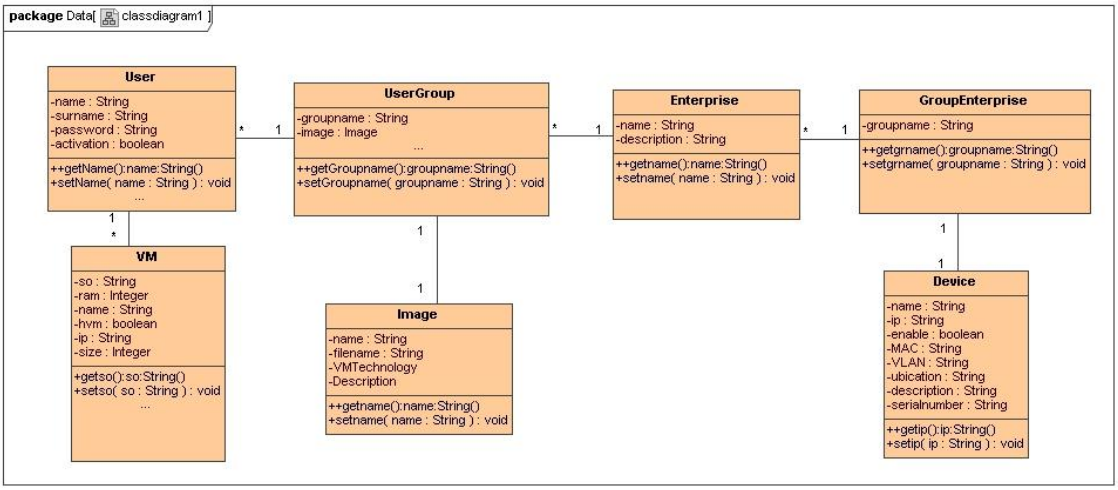

Fig. 2.5 Class diagram

2.4. Information structure diagram

This section shows the information relationship and organization. The use of tree architecture is very suitable. Profiles are defined and then people's organization units are linked with one of them. It is very interesting to define the thin client's devices with all of his properties. In addition, images are also defined in this tree making this architecture very malleable.

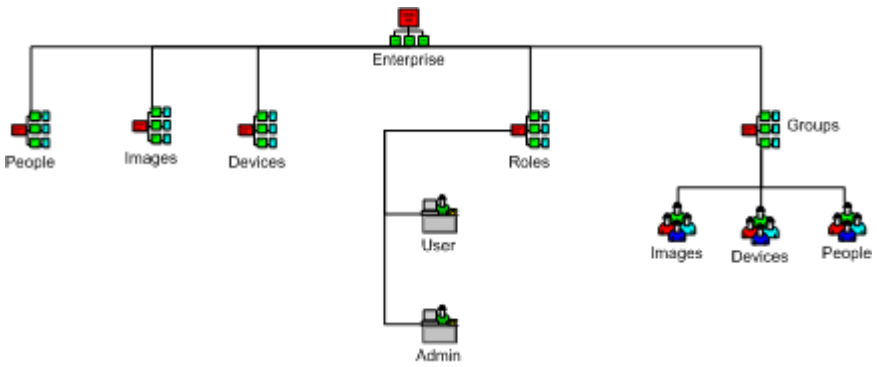

Fig. 2.6 Information structure diagram

2.5. Connection manager activity diagram

Connection manager is accessed by the user authentication component and on first instance it responds on demand for a desktop session requests. It also orders to the virtual machines manager the proper operations depending on the user request.

This is the activity diagram of the connection manager component. In any case, it sends a response to the authentication manager. If a user has an opened session, then it will be opened, it is a new one, then is created the session information to be sent will differ.

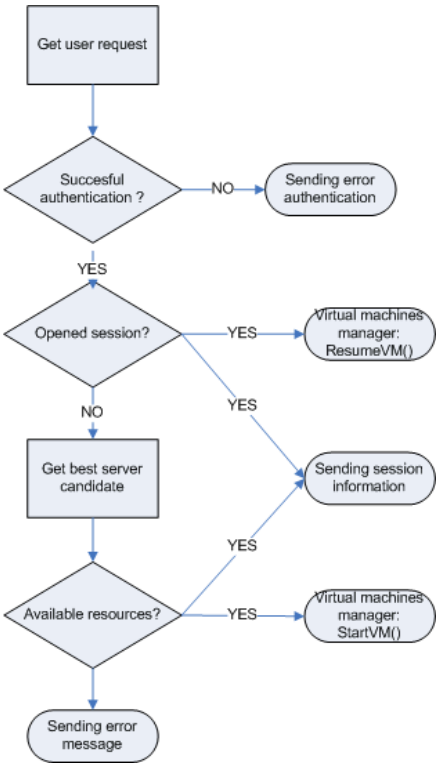

Fig. 2.7 Connection manager activity diagram

2.6. Authentication module computation point of view

This chapter shows the computation point of view of the implemented authentication module, this module also connects with the virtual machine class in order to obtain information parameters like the IP address. Controller and view components classes on the presentation layer have been abbreviated.

A three layer architecture is used, working this way has lots of advantages; one of a better code organization and modularity, developers can split their tasks and the project will be finished combining all the parts.

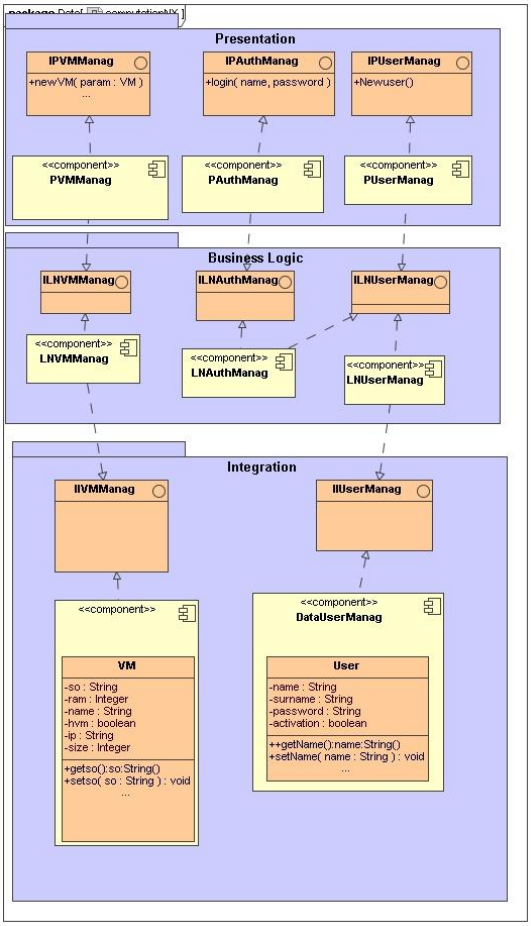

Fig. 2.8 Authentication module: computation point of view

2.7. General computation point of view

Next page shows the full computation point of view. Note that last picture is located at the upper part of the page.

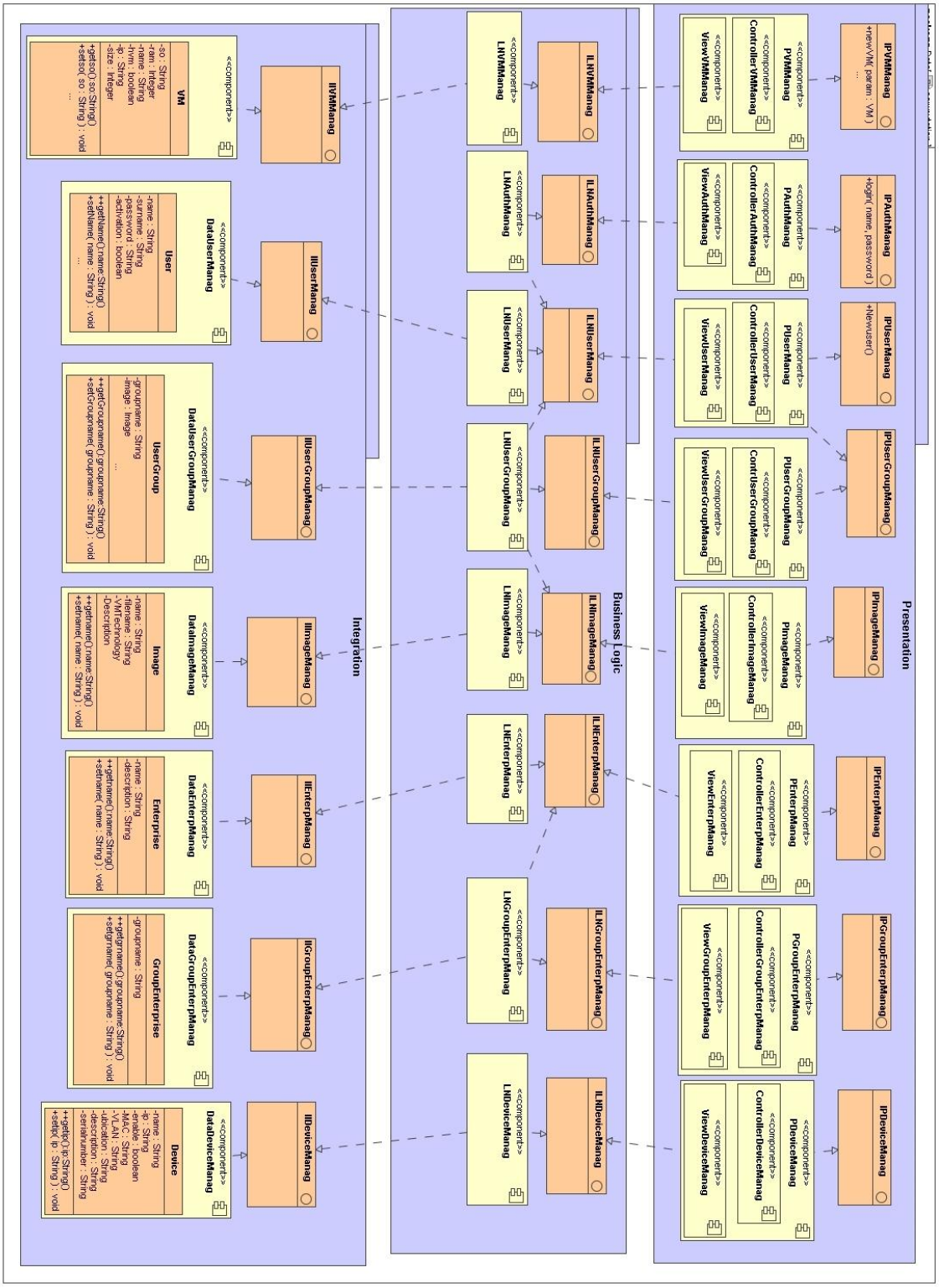

Fig. 2.9 General computation point of view

CHAPTER 3. SYSTEM ARCHITECTURE

This chapter starts defining about candidate technologies and system architecture in order to achieve the requirements defined on the last two chapters.

3.1. The thin client

3.1.1 Connectivity

Most of thin clients are physically situated in foreign offices; this section describes the connectivity parameters between the data centre (where the final operating systems are running) and the user locations.

Due to the need of providing automatic network information (via DHCP) and better network management, remote thin clients devices needs to have level 2 connectivity with the data centre.

There are many ways to connect networks at level 2, for example, using Metro Ethernet over optical link connections. The most usual case will be a remote office with a standard ADSL line, so with two commercial network devices on each network edges will provide desired connectivity. Next figure shows this basic architecture:

Fig 3.1 Tunnel between networks

3.1.2 Remote desktop technologies

There are many technologies that can be used to connect our thin client with his Operating System. This chapter shows the main principles and it justifies the chosen technology.

In one hand we have the KVM over IP technology that works like a standard KVM [3.1] switch but using the IP layer to be used from a remote client. This technology uses a dedicated microcontroller, so it is a full hardware solution; the

problem is that there is a big necessity of network bandwidth that in an operational platform could not be available.

In the other hand, there is an extended list of technologies working on the OSI's application layer. These are the compared technologies and their main characteristics:

- **VNC:** developed by Olivetti & Oracle Research Lab. There are a lot of different implementation and it's very popular, it uses the RFB protocol [3.2]. This protocol it is very simple and works sending the different desktop parts with a compressed image. Nowadays, this technology uses too much network's bandwidth and it do not transmit to the final user the "local" feeling desired.

- **RDP:** Remote Desktop Protocol. Based on the ITU T.share protocol (also known as T.128), this technology was introduced as a service in the early Microsoft Windows NT 4.0. It is very suitable if the offered Operating Systems are Windows based. RDP it is also oriented to give application remote connection, so this option it can be take in consideration.

- **X Window System:** (commonly X11 or X) it is the basic framework to build GUI environments but in this case it was specially designed to be used over the network. So XServer and Xclient can work with independent client-server scenario. Next picture shows how X server takes input from a keyboard and mouse and displays to a screen. A web browser and a terminal emulator run on the user's workstation, and a system updater runs on a remote server but is controlled from the user's machine. Note that the remote application runs just as it would locally.

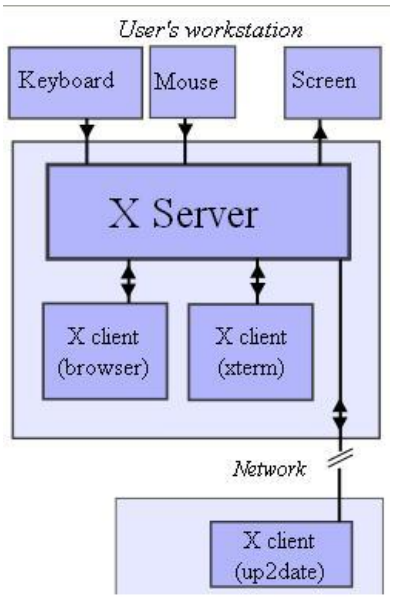

Fig 3.2 The X11 technology

- **NX:** cross-platform technology that uses the X11 protocol making it better compressing and caching data. It also can be used as a proxy-tunnel for RDP connections. So it is very suitable for a platform that wants to offer different types of Operating Systems and their services.

As a conclusion, NX technology represents a good opportunity to have an open start point to work with any thin client. NX is developed by Nomachine [3.3] but this enterprise has opened both server and client as "The FreeNX Project" [3.4] under the GPL license.

Over NX technology the chosen technologies that can work and be very suitable are: X11 in case of using Linux environment and RDP in case of using remote Microsoft Windows desktops and/or applications.

Next figure shows the project's technology map:

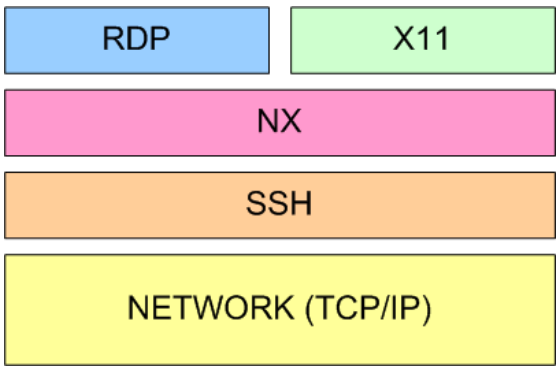

Fig 3.2 Technology map

3.1.3 Hardware device

The present document has been talking about "thin client" or "zero client" in many times; if the client is "thin" or "zero" it's only a way that tries to define the final idea. The idea consists in finding the way to reduce the client device as possible.

Previous chapter (section 3.1.2) talks about using low level protocols like KVM over IP, this option reduces the device complexity but using too much bandwidth resources. Having discarded this option, thin client device will have a minimum hardware elements and an embedded operating system (in some communities, it is known as the firmware).

Exploring on the Internet it can be found two suitable examples:

- **EPATEC 4800:** CPU 1.2 GHz (VIA Eden Esther), 512 MB RAM, 2xUSB 2.0 ports. Ethernet 10/100 Mbps. This device has no hard disk and it does not have fan, so it is extremely quiet. It has the option to boot with DHCP and PXE boot technologies and it can run Windows and Linux. A

Secure Digital memory can be plugged inside and the power consumption is about 20W. [3.5]

- **ThinkWorks:** CPU AMD Geode GX 533. 64 MB FLASH and 128 MB SDRAM. Ethernet 10/100 Mbps. This device has also no hard disk and it offers also both DHCP and PXE boot technologies. [3.6]

Epatec's product will be the chosen due to its suitable features (for example its 512 MB of RAM) and the possibility to connect a Secure Digital memory card. This product has a good reseller near Barcelona and its price is very correct.

Fig. 3.4 Epatec 4800

3.1.4 Embedded Operating System and application

Having chosen the remote connection technology and the hardware thin client device, this section describes the operating system and the applications that will be deployed.
The operating system chosen is Linux, there are a lot of projects that has been studying the way to reduce the distribution size in order to work on small workstations. In this case, this device only needs an Operating System (Debian Linux), a Internet Browser (Mozilla Firefox) and the NX client.

Debian Linux is an extremely tested distribution that will provide the desired stability and usability. Mozilla Firefox will reach the initial requirements expectations working as a start point of the user's session.

As told in the previous chapter, it is very important to have a reliable local system available on the thin client device. So a recovery and update procedure must be used to ensure this. The whole memory card can be parted into 3 partitions:

- **Boot partition:** holds the boot loader information. The boot loader chooses which partition will be loaded, so when a new operating system is deployed into the secondary partition, the boot loader swaps the other partition to boot.

- **Primary partition:** this partition stores the operating system in use

- **Secondary partition:** is used to deploy an updated operating system

Next figure shows the memory card distribution:

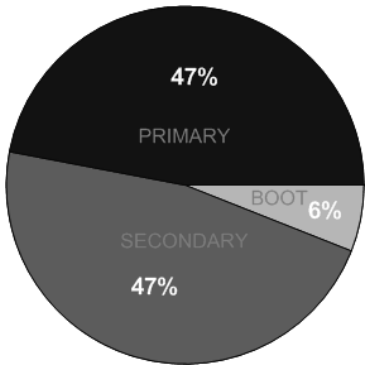

Fig. 3.4 The memory card distribution

3.1.5 NX session authentication

A standard web browser with enabled Java must be installed into the thin client device. In this case Mozilla Firefox and Java Runtime Environment 6 is displayed in full screen mode (using a kiosk add-on), note that only this application is available by the user. Computation point of view is shown on Figure 2.8.

Web browser connects to a web page located into the authentication server and then an authentication form is prompted (Fig. 3.6). When user sends its credentials, the NX client is opened through a web applet (Fig.3.7); NX client waits for a session information file that is dynamically created by the authentication server. Next figure shows the authentication procedure:

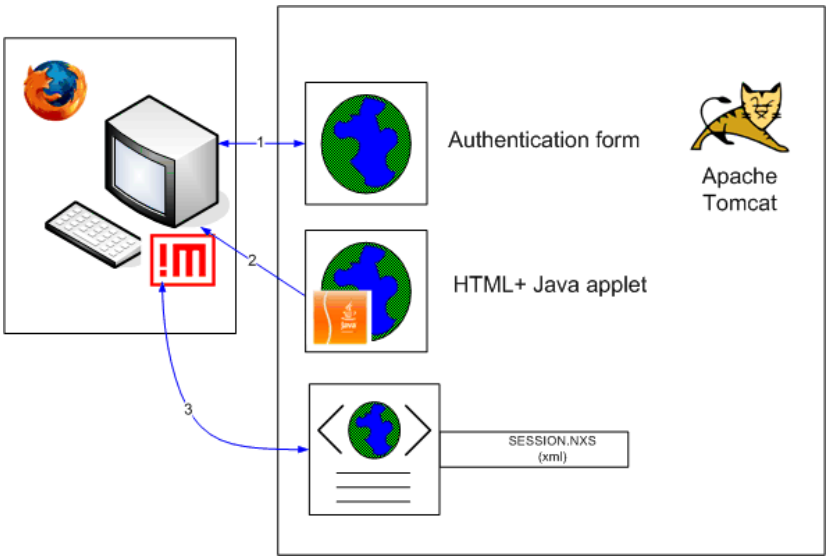

Fig. 3.5 Authentication procedure

All those documents are generated dynamically, so this authentication service is deployed into an Apache Tomcat server, this java application is defined in section 3.2.1.

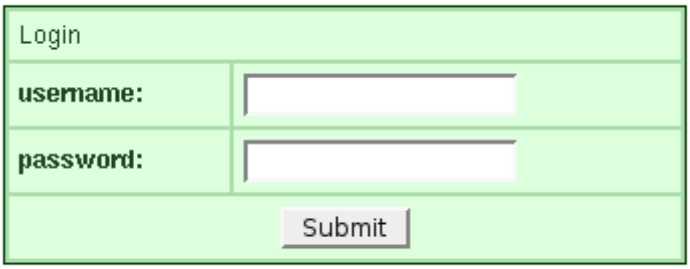

Fig. 3.6 Authentication form

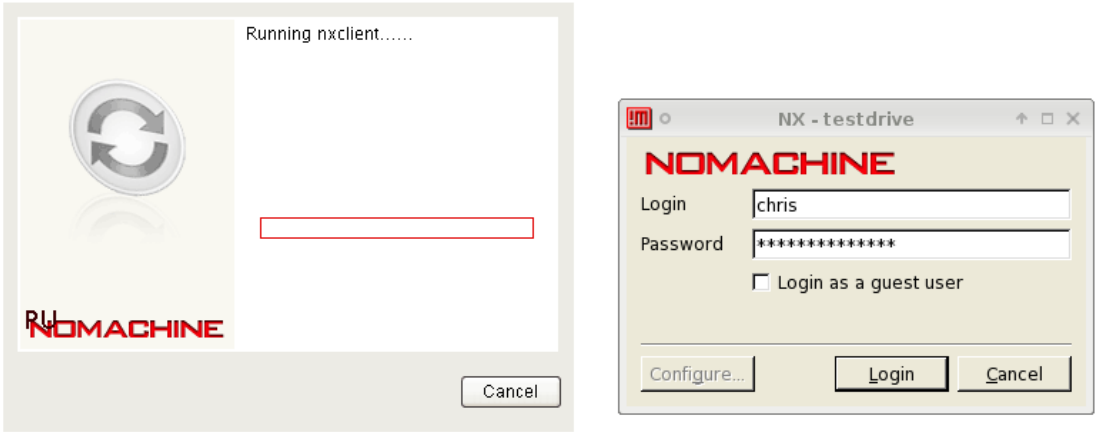

Fig. 3.7 Web applet and NX client

Next figure shows the final result: a working remote desktop on the client display.

Fig. 3.8 Remote desktop

3.1.6 Embedded OS distribution and update

Thin client device boots from network with DHCP and PXE protocol, through a TFTP connection it receives a mini Linux kernel that will mount a remote NFS directory; then rsync utility is used [3.7]

Rsync is a fast incremental file transfer utility, it will synchronize the last update and at the end a script will change the boot loader information.

Local operating system distribution (also called firmware) is not implemented in this project. There are many open-source projects that could help to perform similar operations with similar result. These are some of them:

- **Partimage:** server-client application to copy and backup system images
- **Brutalix:** Red Iris project that tries to compete with IBM's Tivoli Rembo
- **Ploplinux:** Live CD with software deployment utilities

Last two utilities uses Partimage utility [3.8]

Note that this procedure is also used to easily deploy lots of thin client devices.

3.2. Virtual remote desktops administration

I2Cat foundation developed a project called Virtmanager. The project consisted in developing a web application that could manage virtual machines using Xen virtual technology. With this application, it is possible to create, delete, stop and pause pre-installed Linux or Windows virtual machines.

Main platform manager functional requirements are solved with this application, but it must to be integrated to this new project.
These are the needed changes that must be done to the actual I2CAT's virtmanager project:

- Virtmanager database integration: the application needs to work with a persistence database.
- NX for virtual machines administration: an IT operator/administrator should access to a existing desktop session
- Virtual web services migration: it must split the virtual invocation method to a generic interface via web services, the virtual machines manager will be the result.
- Load balancer implementation: needed to achieve an scalable and low cost architecture
- User profiling implementation: ACL implementation

Next section explains the different challenges to be faced with I2CAT Virtmanager project.

3.2.1. NX integration

This section describes a Virtmanager implemented feature. This application can create, pause and delete virtual machines but there is no possibility to access to its desktop session. A user must connects manually using an RDP o X11 client.

The main objective is to provide to the Virtmanager users the possibility to connect to the virtual machine desktop environment. So the same technology explained on section 3.1.5 is used: a Java applet that executes the NX client and a dynamic NX session file.

The starting version of virtmanager does not has persistence and some virtual machines values (such IP address) needs to be stored on somewhere. So Machine class has been integrated to a MySQL database, this process it is shown on section 3.2.2.

Next figure shows the new X11 icon as with the new added feature:

Fig. 3.9 NX Virtmanager integration

3.2.2. Platform persistence

Platform persistence is used to store information on a database.

There are two information groups:

- **User and device information:** section 2.3 shows the class diagram and section 2.4 shows an information tree.

- **Resource information:** the resource monitoring tool described in section 3.2.5 stores the virtual pool machines system load. This information is used by:

    - Connection manager: performing load balancing tasks and user session assignments

    - Platform manager: performing virtual machines administration and control tasks. Also useful to have a statistical control.

In the first case, the use of an LDAP service could be interesting, users and even thin clients devices could be stored with an LDAP hierarchy. Of course, the use of a relational database could be use instead; LDAP can provide to other platform services (like samba file sharing) an authentication layer.

On the second case, MySQL server is a good candidate to store a big amount of monitoring data and also virtual machines information.

From business logic level, a platform application could access to the proper integration class. LDAP access can be performed through a JNDI [3.9] connector and resource information can access to a MySQL server with the Java Hibernate API as an implementation of the Java Persistence API [3.10].

Java Persistence API (JPA) allows managing relational data from the Java platform; Hibernate is a JPA framework and is used into this module. Hibernate maps relational databases with an object-oriented model.

3.2.3. Virtual machine technology: XEN

Xen is a virtual machine software and Xen-source is the open-source project. It is developed thanks to the open source community from the first original idea written by Ian Pratt.

The hypervisor is the lowest layer and it permits to execute multiple guest operating systems. Dom0 is the initial guest operating system and DomU are the other multiple guests. Next figure shows the Xen basic architecture:

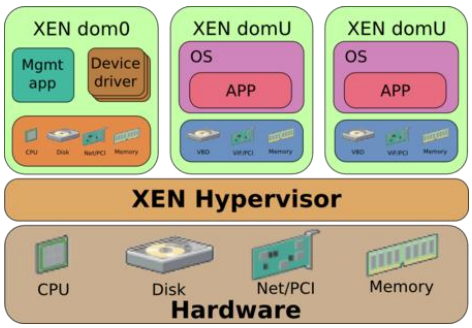

Fig. 3.9 Xen basic architecture

These are the main features supported by the last version:

- Paravirtualitzation support
- Native virtualization support
- Real time Virtual machines migration
- Improved performance

3.2.4. Virtual machines web services

One of the functional requirements (section 1.2.4) is that virtual machines manager has to be capable to work with different virtualization technologies.

This component is not developed on this project; this is an approximation of how it would be. The deployed technology is Xen, but keeping in mind future technologies could appear in the future.

Virtual machines manager should be implemented as a web service [3.11].

A web service is defined by W3C [3.12] as “a software system designed to support interoperable machine-to-machine interaction over a network”. The first basic idea is to communicate from client to a server XML over HTTP, all of this following the SOAP standard [3.13].

Platform manager or connection manager could receive operation request. For example, if a virtual machine called “machine01” running with Xen technology has to be paused, a request with the action “pause” can be sent and a translation interface could send to the application layer: “xm save machine01”.

Virtual technologies are evolving very fast and this system must to be prepared for constant upgrades to different virtualization platforms.

Web services are defined with generic operations: startVM(), stopVM(), pauseVM() etc.

So this component is able to send to the respective technology component, the proper operation. In this example, there is a Xen [3.14] component and a VMware component [3.15].

Next figure shows the implementation diagram of this part:

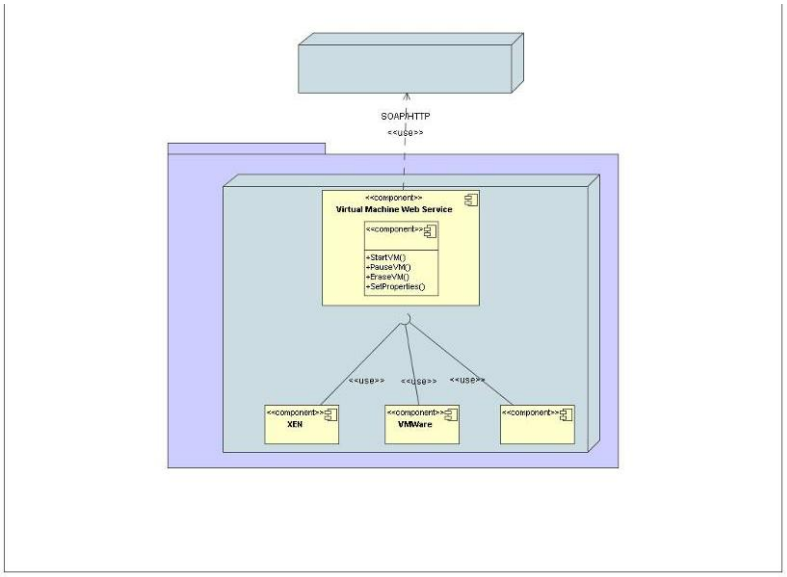

Fig. 3.10 Virtual machines manager web service implementation

3.2.5. Resource monitoring

Resource monitoring is needed to:

- Optimize system resources
- Provide an acceptable desktop session quality of service

Virtual machines subsystem could be a cluster of servers that are able to run instance machines; every instance needs to be provided by a minimum resource requirements. So the best way to insure it is to monitor all the server's resources.

All the servers need to have installed agents that will response to monitoring incoming requests from the master monitor. Monitor node will store the results to a database.

Nowadays there are two popular monitoring tools:

- **Nagios:** Web site quote: "Nagios is a host and service monitor designed to inform you of network problems before your clients, end-users or managers do"
- **Zabbix:** Web site quote: "ZABBIX offers advanced monitoring, alerting and visualisation features today which are missing in other monitoring systems, even some of the best commercial ones"

It seems that both tools are more or less the same, but in our case, statistical service information is needed. Zabbix provides better resource statistics than Nagios because this last one is more specialized in availability. In addition, Zabbix offers the possibility to use proxy agents (very useful for a physically distributed infrastructure).

So monitoring resource information is stored on a MySQL server, this information is used by Zabbix tool as system monitoring service and also by the connection manager as a load balancer.

3.2.6. Access Control Layer

On different parts of the project, authentication and user profiling must to be accessed. So the best way is to centralize this requisite to a directory service.

OpenLDAP is an open-source directory service for user and (in this case) device authentication, is much extended and very stable; it is also used in this project to manage the different desktop images. OpenLDAP can be patched to work with DHCP3 server performing device management from a single service.

So a user is linked to a profile and it has access to a list of desktop images. For example, a user could not use an image with development software. The association of users, devices and images it is very useful and makes easier the administration.

User profiling it becomes very useful creating different permission layers in case of complex and big enterprises. The creation of different profiles permits the tasks definition permitted by each user. For example: to reset virtual machines or even create new ones; or in the other hand, user profiles with nearly no permissions.

3.2.7. Platform manager: Virtmanager

There are lots of technologies that could be used to develop a web application as the platform manager, but it must be kept in mind that this component needs to interact with other heterogeneous services; so it could be based on modular components and running into a stable application server.

Java Platform, Enterprise Edition (or Java EE) is a platform for server programming based on Java that can face all the required challenges. In addition, its modularity and its continuous evolution make it suitable for future proposals.

As said in section 3.2 introduction, virtmanager is a start point application developed by I2cat foundation. Its modularity permits to perform all the described tasks with no significant code change. Model-View-Controller architecture is used in this application, in concrete Apache Struts.

Apache Struts is comprised of a controller servlet, beans and other Java classes, configuration files, and tag libraries. Architecture is shown on next figure:

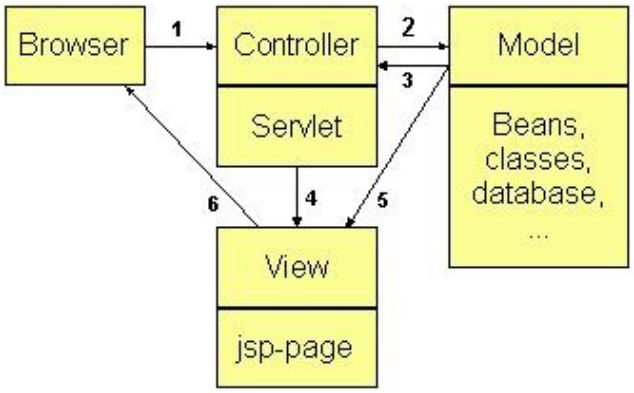

Fig. 3.11 MVC architecture

3.2.8. Connection manager

It must keep in mind that if an enterprise wants to offer this product to a large number of users, the server infrastructure needed will grow very fast. So it is very important the system resource optimization.

There are multiple load balancer implementations for various types of clusters, but in this case, this server cluster is very specific and it could be implemented.

Section 3.2.6 justifies the use of JavaEE, so if user authentication and platform manager components are integrated into a java application server, connection manager module it could be also implemented with this technology. JavaEE modularity will make easier the implementation.

Connection manager is accessed by the user authentication component and it responds to session information requests. At the end of this section, the connection manager algorithm is shown. These are the information sources:

- **Cluster resources:** connection manager wants to access to the cluster resources, it could access through an integration class using the Hibernate API; but using Zabbix tool monitoring, it is interesting to use the “Zapcat” API [3.13].

  As shown on next figure, Zapcat tries to straddle the Java world (in this case JavaEE) and the “sysadmin” world (cluster server). On one end, it is a simple Java object that uses standard Java API’s and blends well into web applications. On the other end it speaks the protocols that Zabbix uses to gather statically data. On top of Zapcat sits JMX [3.19] as a common language to speak.

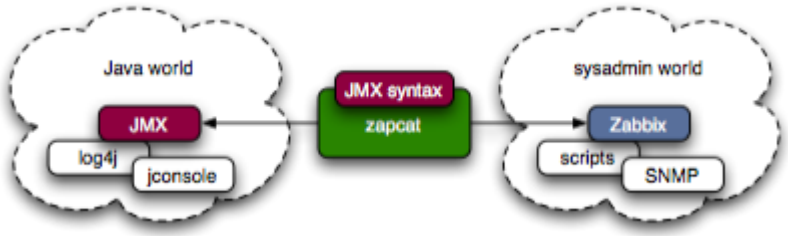

Fig. 3.12 Zapcat situation

- **Existing user session:** users may already have an opened session; this information is stored into the user and information module described on section 3.2.2. When connection manager consults the user information, the component knows if there is an opened user session. JNDI connector is used to access to the LDAP server and integration classes are developed into the integration layer.

With this two source information, session information is generated and sent as a response to the user authentication and it sends the proper action to the virtual machines manager component. Figure 2.7 is the activity diagram that shows the connection manager algorithm.

3.2.9. Storage

This section talks about the storage system. It is very interesting to provide a unique point of storage where all the components are stored; saying that is unique does not means that high available and redundant techniques are omitted.

A Storage Area Network [3.20] could be the best choice.

For a high demand platform: switched fibre channel architecture or iSCSI [3.21] could be necessary. Next figure could be an option:

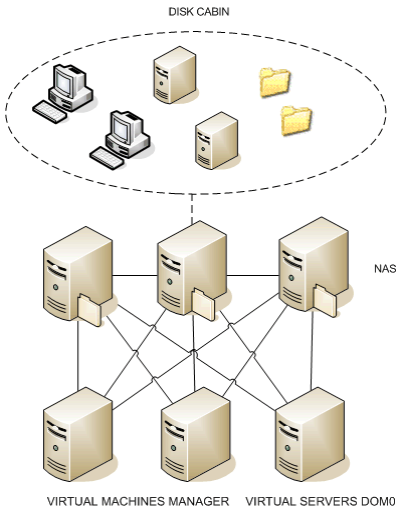

Fig. 3.13 Storage architecture

Virtual machines are stored and accessed by the cluster server dedicated to provide desktop sessions.

All the explained services during this section should be installed as virtual machines, all of them stored into the storage system. Server virtual machine can be used on any other machine with hardware independent configuration; it will help on disaster recovery and new service deployment.

For a low demand platform, a normal server with high hard disk capacity (or connected to a storage cabin) serving with Network File System [3.16] or Lustre File System technology [3.17] should be enough.

This last option is cheaper than the first one, Fibre channel hardware is quiet expensive.

As a conclusion, these are the main components to store:

- User, group and enterprise files and directories
- Desktop virtual machines: operating system instances.
- Server virtual machines: platform servers are virtualized and its file system is stored into the main storage system.

3.2.10. Virtual machines cluster

The purpose of this section is to define the best architecture to provide scalability with the minimum operational cost. The second main objective is to add easily new server machines when the number of users increases.

There are lots of cluster architectures and proposals. One of the best and popular is what is called a “Beowulf cluster” [3.22] [1].

A common operating system is shared from a disk cabin or a NFS server, so cluster nodes boots from network and hard disk is not needed. If the common operating system is modified, all the cluster nodes are modified and new nodes addition results a very easy procedure.

Next figure shows the cluster architecture:

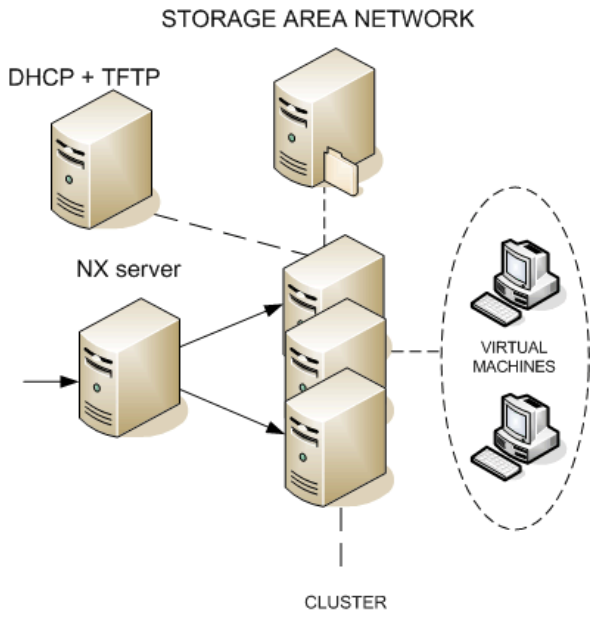

Fig. 3.14 Cluster architecture

3.3. Global system proposal

3.3.1. Architecture proposal

Next figure is the final architecture proposal with the justified and explained technologies during last section of this chapter; note that connection technologies between nodes are also defined.

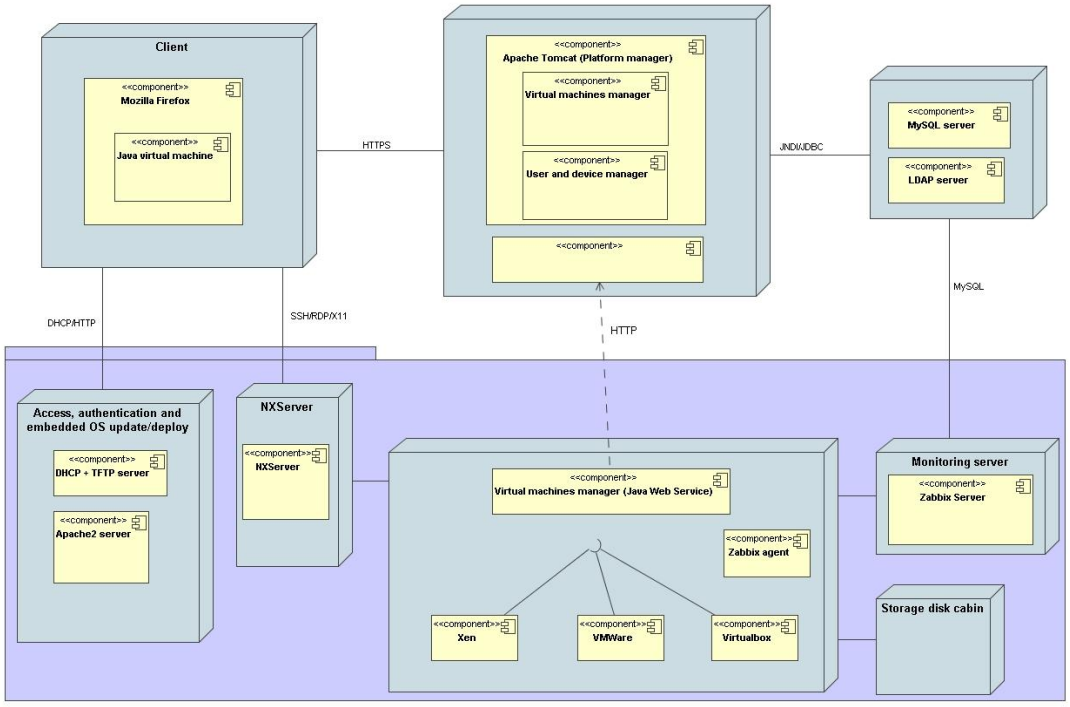

Fig. 3.15 Architecture proposal

3.3.2. Operational proposal

Next figure shows the operational proposal architecture. Based on the last picture, this is a basic topology idea. Note that network topology could have other valid variations.

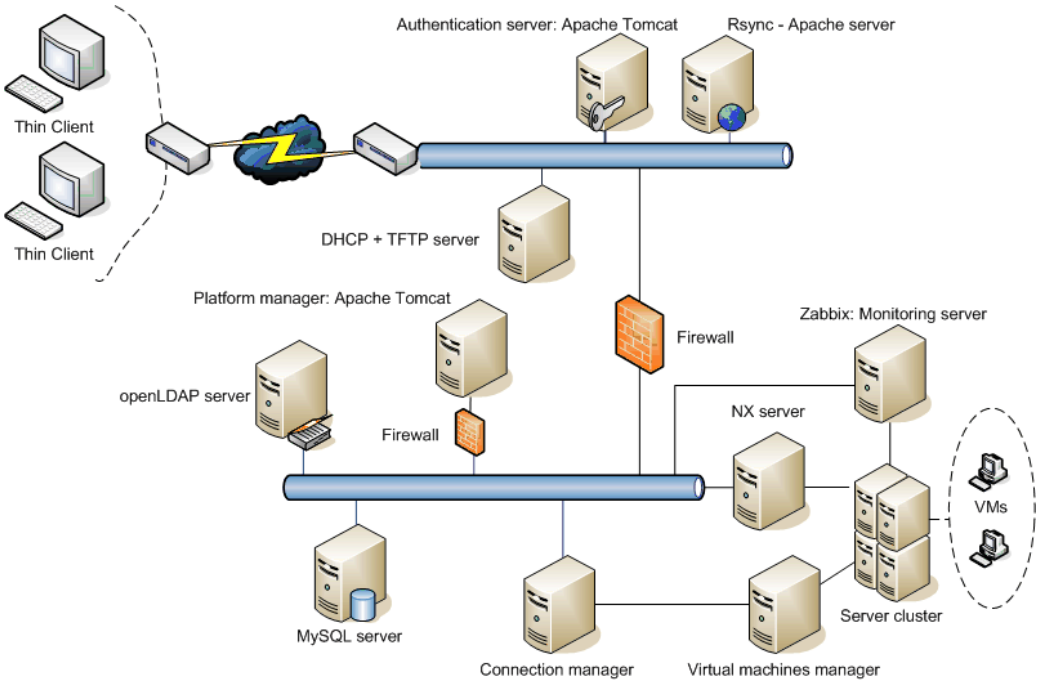

Fig. 3.16 Operational proposal

3.3.3. Virtualized architecture proposal

This project explains in many times the advantages using virtualization, so it could be very interesting to apply this technology to the service infrastructure. Next figure shows a possible architecture using virtualization technology.

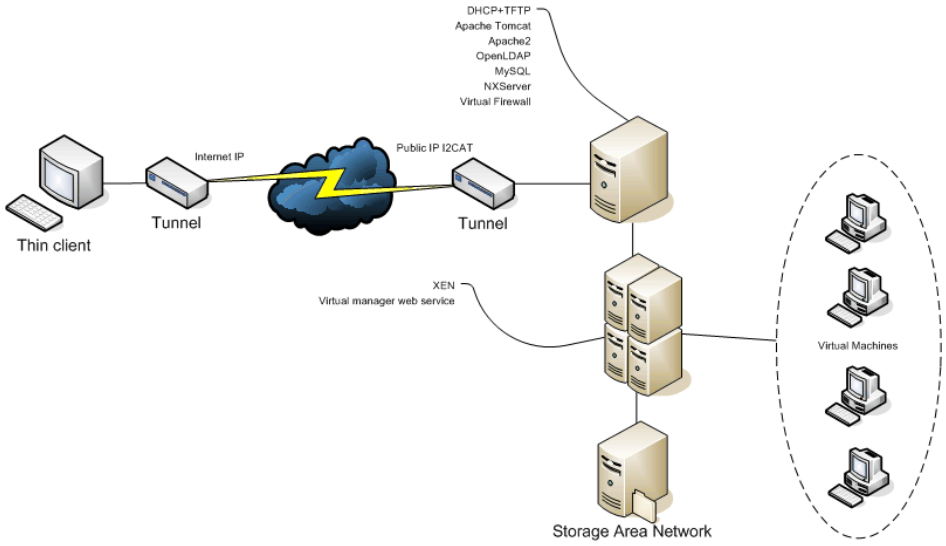

Fig. 3.17 Virtualized architecture proposal

Last picture shows how with one server, with a Xen patched Linux kernel, it can virtualizes multiple instances with different services installed; there are virtual machines that acts as firewall between the others VMs, so a virtualized network

can be deployed with one physical server. Every virtual machine is independent with rest; it is a security contribution to keep in mind and provides improved Mean Time to Recovery (MTTR).

3.4. Used tools and technologies

This section describes the used tools during the project realization.

3.4.1. Server technologies

- **Apache2:** web server, an rsync client can download embedded operating system updates
- **MySQL:** database to store the cluster resources state
- **Zabbix:** server and agent for resources monitoring
- **DHCP and TFTP:** gives network parameters and sends a minimal linux kernel
- **Java Web Services:** remote method calls and message parsing for virtual machines management
- **OpenLDAP:** open-source directory service for user and device authentication
- **Apache Tomcat:** Java servlet container and also serves jsp pages
- **FreeNX:** open source implementation of NX server used to connect to remote desktops

3.4.2. Application and tools

- **Hibernate API:** java-based object-relational mapping/persistence framework
- **Junit API:** testing framework for Java
- **Struts1/2 framework:** open-source application framework for developing Java EE web applications

3.4.3. Client applications

- **NX client:** client to connect to the freenx server.
- **Mozilla Firefox:** web client with java plugin to execute applets.
- **Java Runtime Environment:** java virtual machine used to execute applet from Mozilla Firefox

CHAPTER 4. BASIC MARKET ANALYSIS

4.1. Product description

This chapter tries to define a basic business case of this project. Potential clients interested on this product does not want only to cover a part of they IT necessities, they want an entire integrated solution.

The integrated solution could be:

- Thin client devices under a renting or leasing contract
- Remote desktops and applications
- Network connectivity between networks and Internet
- Office software suite integration (ERP)

4.2. Market targets

The main detected targets are:

- **Small and medium business:** this target is very large. We also should focus with new enterprises because they does not know if they will survive after a year, so this product seems very suitable for them.
- **Isolated public administrations:** country side city halls have only one or two computers, so this solution could cover an uncovered necessity.
- **Educational centres:** intermediate and universities centres could be interested with this project. Those targets are more difficult because those centres are used to implement their own services.

4.3. Product added value and weak points

These are the characteristics that improve the added value of this product:

- The main targets are small and medium business but without forget big enterprises
- The capacity to fix incidents remotely permits an efficient service providing reliability and availability to the user.

- Physical installation value is lower at the client side. No matter if a device is stolen.

- The classic initial IT investment for a new business it is very annoying, this solution contributes to make this deployment easier.

- The full solution also offers data and voice services; the client does not have to worry about this.

Weak points that affect the product are:

- Human security and privacy suspicion due to the fact that information is stored remotely

- Technological limitations: If a remote client has lots of devices, it will be needed a high capacity data connection. This is not always available.

- Connection dependent platform: If connection link is down, thin clients won't connect to the remote desktops and it will affect the productivity.

These are the possible solutions to the weak points:

- Potential clients must trust with the project and maybe the best way could be to show the system in deep. In addition, it could be recommended to offer trial periods to show the benefits of the platform.

- The second and the third point can't be solved, but fortunately, networks are getting faster and more reliable.

4.4. Competitors analysis

Nowadays it seems that the topic "virtualization" is getting popular. Even the marketing people link this concept with lower environmental impact.

Citrix Systems is one of the first companies that bet for the idea of remote desktop and applications. They have similar products but only affordable for big accounts.

VMWARE was one of the first companies that started to sell virtual platforms. They also offer remote virtual desktops but also orientated to big accounts.

As a conclusion, there are some big enterprises that offer similar products but only for big accounts. So it seems that it is a market gap that is not covered, the small and medium business of more or less 10 workers.

CHAPTER 5. FUTURE PROPOSALS

The project main objective is to define an optimum and scalable platform that can provide virtual remote desktops. Some parts of the project have been developed, but it remains most of the functionalities to implement.

These are the main future proposals to implement:

- **Platform management:** the platform manager should be implemented to control: users and virtual desktops.

- **Connection manager and desktops cluster:** it must be implemented this module to perform load balancing tasks and session manager, all of it interacting with a cluster of servers providing virtual desktops.

- **Virtual machines web services:** an interface via web services should be implemented with the possibility to introduce newer virtual platforms on the future.

- **Resource monitoring service:** with a running Zabbix server, connection manager can connect with the Zapcat API performing load balancing tasks. It is also accessed by the platform manager to analyse the use of the system.

- **Virtual machines migration:** it could be very interesting to use the migration processes that permit to move a virtual machine between servers with no user knowledge.

- **Embedded operating system management:** it should deploy the infrastructure that deploys embedded operating system updates into the thin client devices.

CHAPTER 6. ENVIRONMENTAL IMPACT

Lots of students have a lot of problems writing this chapter, depending on the project, it is quite difficult to define the project environmental impact.

This project can reduce the environmental impact with: power consumption and electronics waste reduction.

- **Power consumption:** next table is the power consumption between: a standard computer and the needed elements to run a desktop session service.

Standard computer	
Device	**Consumption [W]**
Computer	200
Total Consum.	200

Thin client architecture	
Device	**Consumption [W]**
Computer	20
Server	81,25
AC (COP 3.0)	27,08333333
Total Consum.	128,3333333

Table 6.1 Power consumption comparison

Server calculations are based on a power consumption of 650W where a minimum of 8 sessions can be executed on each one. European Energy Star program [5.1] recommends the use of COP 3.0 to calculate the air conditioned cost (total device consumption / 3).

- **Electronics waste:** reductions on electronics waste are very important. Thin client devices has less electronics components, for example, note that it does not has the typical hard disk with only a minimum percentage usage.

As a conclusion, thin client architecture consumes 36% less than architecture based on standard computers. Reduced operative cost and electronics waste means that this architecture respects the environment.

CHAPTER 7. PROJECT TIME PLANIFICATION

The history of this project began when I2CAT foundation was asked by an enterprise to develop a small model with thin clients. During the functional requirements definition, the enterprise was gone but the project did not stop.

The project has two big stages: the first one correspond to the definition and specification and the second one it corresponds to the implementation of some modules. The first one it is the most important, if the specifications are well defined, a group of developers could implement the entire service in just a few months.

Next figure shows the Gantt diagram, it can be seen that the main goal of this project has been the design and definition of an entire virtual platform interacting with thin clients.

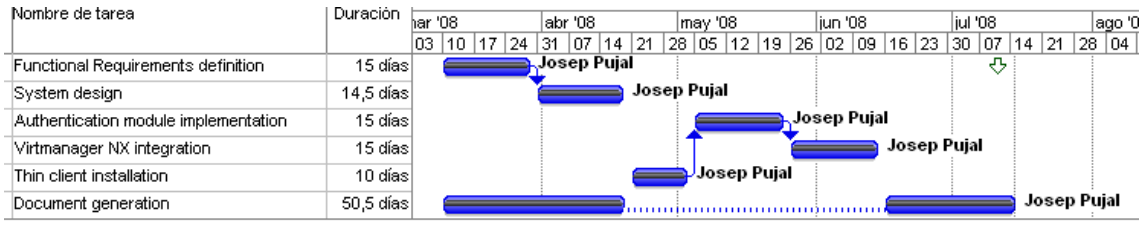

Fig. 7.1 Gantt diagram

CHAPTER 8. CONCLUSIONS

Most of the functional requirements of this project have been achieved; the study of the different involved technologies has demonstrated the viability of the entire project. Implemented modules represent a final product's probe of concept for a big institution or business.

This document is an important part of a big project. The document it represents the specification, the analysis and the design of a platform that could be applied to different working environment. Educational institutions, such a university, could be interested to provide to his students remote licensed applications or laboratory desktops. Enterprises with physically distributed offices can reduce operational costs such support services, software deployment and even power budget. In addition, the possibility for a user to work wherever he wants with his remote personal desktop will represent better productivity.

Virtual technologies are getting more famous and big software solutions enterprises are developing similar products; this project has demonstrated that with no big economic investments, an operating platform could be working.

REFERENCES

[2.1] http://www.uml.org
[2.2] http://www.rm-odp.net
[3.1] http://en.wikipedia.org/wiki/KVM_Switch
[3.2] http://www.realvnc.com/docs/rfbproto.pdf
[3.3] http://www.nomachine.com/
[3.4] http://freenx.berlios.de/
[3.5] http://www.epatec.es
[3.6] http://www.applica.com/products/ThinWorks.htm
[3.7] http://samba.anu.edu.au/rsync/
[3.8] http://www.plop.at/en/ploplinux.html
[3.9] http://java.sun.com/products/jndi/
[3.10] http://www.hibernate.org/
[3.11] http://en.wikipedia.org/wiki/Web_service
[3.12] http://www.w3.org/
[3.13] http://www.w3.org/2000/xp/Group/
[3.14] http://www.cl.cam.ac.uk/research/srg/netos/xen/
[3.15] http://www.vmware.com/
[3.16] http://nfs.sourceforge.net/
[3.17] http://www.lustre.org/
[3.18] http://www.kjkoster.org/zapcat/Why_Zapcat.html
[3.19] http://es.wikipedia.org/wiki/JMX
[3.20] http://www.rediris.es/rediris/boletin/58-59/ponencia9.html
[3.21] http://tools.ietf.org/html/rfc3720
[3.22] http://www.beowulf.org/
[5.1] http://www.eu-energystar.org/es/es_calculator.shtml

BIBLIOGRAPHY

[1] **Sloan, Joseph D.** Linux Clusters with OSCAR, Rocks, openmosix and MPI. O'Reilly Media. 2005

INDEX OF PICTURES

ANNEX A: Thin client installation

A network installation will be the best way to make the initial image of a thin client. When the image is done, it will be easy to clone dozens of thin client devices. This is the configuration to be done from a computer next to the thin client device.

TFTP server installation:

> apt-get install tftpd-hpa
> vi /etc/inetd.conf

tftp dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /tftpboot

DHCP server installation and config:

> apt-get install dhcp3
> vi /etc/dhcp3/dhcpd.conf

```
subnet 192.168.5.0 netmask 255.255.255.0 {
        option domain-name-servers 62.204.192.20, 192.168.1.24;
        option subnet-mask 255.255.255.0;
        option routers 192.168.5.1;
        next-server 192.168.5.1;
        use-host-decl-names     on;

        group {
                #new_box
                host new_box {
                        filename                "pxelinux.0";
                        hardware ethernet 00:d0:b7:16:62:53;
                        fixed-address 192.168.5.12;
                }
        }
}
```

(write the proper MAC and IP)

Boot software with PXE:

> apt-get install syslinux
> mkdir –p /tftpboot/pxelinux.cfg
> cp /usr/lib/syslinux/pxelinux.0 /tftpboot/

A Debian etch will be installed, it is needed to copy some distribution files on the tftpboot directory:

> mkdir –p /tftpboot/
> cd /tftpboot/

> wget http://ftp.rediris.es/debian/dists/etch/main/installer-386/current/images/netboot/debian-installer/i386/linux

> wget http://ftp.rediris.es/debian/dists/etch/main/installer-i386/current/images/netboot/debian-installer/i386/initrd.gz

> vim /tftpboot/boot.txt

--Boot Menu—
etch_i386_install

> vim pxelinux.cfg/default

```
DISPLAY boot.txt
PROMPT 1
TIMEOUT 60
DEFAULT etch_install

LABEL etch_install
kernel linux
append vga=normal initrd=initrd.gz  --
```

Finally switch on the thin client device, after DHCP negotiation the device will receive the files to perform a normal Debian installation.

A minimal Debian installation must to be done and we will install manually the desired packages.

With a minimal installation these are the steps to install the necessary packages:

> apt-get install ssh iproute
> apt-get install x-window-system-core
> apt-get install icewm
> apt-get install fbpanel
> apt-get install sudo
> apt-get install hsetroot
> apt-get install libaudiofile0

Create a user called “thinuser”

Insert “startx” at the end of thinuser’s .bash_profile

> vi /home/thinuser/.Xsession

```
#!/bin/sh
/usr/bin/icewm &
```

/usr/bin/hsetroot -tile /home/dude/participants5-2.jpg &
/usr/bin/iceweasel http://toru.upc.es:8888/NX/ &
/usr/bin/fbpanel

> vim /home/thinuser/.fbpanel/default

```
Global {
   edge = bottom
   allign = right
   margin = 0
   widthtype = request
   width = 200
   height = 36
   transparent = true
   tintcolor = #000000
   alpha = 125
   setdocktype = false
   setpartialstrut = true
}
```

IceWM configuration files are needed to hide for example window titles and avoid the use of Alt+tab hot key.

.icewm/preferences:

```
#preferences(1.0.0) - generated by genpref

#NOTE: All settings are commented out by default, be sure to uncomment them if you change them!

#  Focus windows by clicking
ClickToFocus=0 # 0/1

#  Raise windows when focused
RaiseOnFocus=0 # 0/1

#  Focus window when client area clicked
FocusOnClickClient=0 # 0/1

#  Raise window when client area clicked
RaiseOnClickClient=0 # 0/1

#  Raise window when title bar is clicked
RaiseOnClickTitleBar=0 # 0/1

#  Raise window when frame button is clicked
RaiseOnClickButton=0 # 0/1

#  Raise window when frame border is clicked
RaiseOnClickFrame=0 # 0/1
```

```
#  Pass focusing click on client area to client
# PassFirstClickToClient=1 # 0/1

#  Focus normal window when initially mapped
# FocusOnMap=1 # 0/1

#  Focus dialog window when initially mapped
# FocusOnMapTransient=1 # 0/1

#  Focus dialog window when initially mapped only if parent frame focused
# FocusOnMapTransientActive=1 # 0/1

#  Colormap focus follows pointer
# PointerColormap=1 # 0/1

#  Limit initial size of windows to screen
# LimitSize=1 # 0/1

#  Limit initial position of windows to screen
# LimitPosition=1 # 0/1

#  Maximized windows can be resized
# SizeMaximized=0 # 0/1

#  Show position status window during move/resize
# ShowMoveSizeStatus=1 # 0/1

#  Display mini-icons on desktop for minimized windows
 MinimizeToDesktop=0 # 0/1

#  Always maintain focus under mouse window (makes some keyboard support
non-functional or unreliable)
# StrongPointerFocus=0 # 0/1

#  Opaque window move
# OpaqueMove=1 # 0/1

#  Opaque window resize
# OpaqueResize=1 # 0/1

#  Windows initially placed manually by user
ManualPlacement=0 # 0/1

#  Smart window placement (minimal overlap)
# SmartPlacement=1 # 0/1

#  Center dialogs on owner window
# CenterTransientsOnOwner=1 # 0/1
```

```
#  Menus track mouse even with no mouse buttons held
MenuMouseTracking=1 # 0/1

#  Auto raise windows after delay
# AutoRaise=0 # 0/1

#  Delay pointer focusing when mouse moves
# DelayPointerFocus=0 # 0/1

#  Support win95 keyboard keys (Penguin/Meta/Win_L,R shows menu)
Win95Keys=0 # 0/1

#  Treat Penguin/Meta/Win modifier as Ctrl+Alt
ModMetaIsCtrlAlt=1 # 0/1

#  Support mouse wheel
UseMouseWheel=1 # 0/1

#  Show popup menus above mouse pointer
# ShowPopupsAbovePointer=0 # 0/1

#  Send the clicks outside menus to target window
# ReplayMenuCancelClick=0 # 0/1

#  Alt+Tab window switching
QuickSwitch=0 # 0/1

#  Alt+Tab to minimized windows
# QuickSwitchToMinimized=0 # 0/1

#  Alt+Tab to hidden windows
# QuickSwitchToHidden=0 # 0/1

#  Alt+Tab to windows on other workspaces
# QuickSwitchToAllWorkspaces=0 # 0/1

#  Manage root window (EXPERIMENTAL - normally enabled!)
# GrabRootWindow= # 0/1

#  Snap to nearest screen edge/window when moving windows
# SnapMove=1 # 0/1

#  Workspace switches by moving mouse to left/right screen edge
# EdgeSwitch=0 # 0/1

#  Display desktop background centered and not tiled
# DesktopBackgroundCenter=0 # 0/1

#  Reload menu files automatically
# AutoReloadMenus=1 # 0/1
```

```
#  Show application icon over menu button
# ShowMenuButtonIcon=1 # 0/1

#  Automatically disable some functionality when running under GNOME.
AutoDetectGNOME=0 # 0/1

#  Show task bar
 ShowTaskBar=0 # 0/1

#  Task bar at top of the screen
# TaskBarAtTop=0 # 0/1

#  Auto hide task bar after delay
# TaskBarAutoHide=0 # 0/1

#  Show clock on task bar
 TaskBarShowClock=0 # 0/1

#  Show APM status on task bar
# TaskBarShowAPMStatus=0 # 0/1

#  Task bar clock uses nice pixmapped LCD display
TaskBarClockLeds=0 # 0/1

#  Show mailbox status on task bar
TaskBarShowMailboxStatus=0 # 0/1

#  Beep when new mail arrives
# TaskBarMailboxStatusBeepOnNewMail=0 # 0/1

#  Count messages in mailbox
TaskBarMailboxStatusCountMessages=1 # 0/1

#  Show workspace switching buttons on task bar
TaskBarShowWorkspaces=0 # 0/1

#  Show windows on the taskbar
TaskBarShowWindows=0 # 0/1

#  Show windows from all workspaces on task bar
# TaskBarShowAllWindows=0 # 0/1

#  Show 'Start' menu on task bar
TaskBarShowStartMenu=0 # 0/1

#  Show 'window list' menu on task bar
TaskBarShowWindowListMenu=0 # 0/1

#  Show CPU status on task bar (Linux & Solaris)
```

```
TaskBarShowCPUStatus=0 # 0/1

#  Show network status on task bar (Linux only)
TaskBarShowNetStatus=0 # 0/1

#  Use double-height task bar
TaskBarDoubleHeight=0 # 0/1

#  Move mouse when doing focusing in pointer focus mode
# WarpPointer=0 # 0/1

#  Allow mouse actions on client windows (buggy with some programs)
# ClientWindowMouseActions=1 # 0/1

#  Draw window title centered
# TitleBarCentered=0 # 0/1

#  Show themes submenu
ShowThemesMenu=0 # 0/1

#  Confirm logout
# ConfirmLogout=1 # 0/1


# Horizontal window border
BorderSizeX=0 # [0-128]

# Vertical window border
BorderSizeY=0 # [0-128]

# Horizontal dialog window border
DlgBorderSizeX=0 # [0-128]

# Vertical dialog window border
DlgBorderSizeY=0 # [0-128]

# Title bar height
TitleBarHeight=0 # [0-128]

# Resize corner width
CornerSizeX=0 # [0-64]

# Resize corner height
CornerSizeY=0 # [0-64]

# Pointer motion distance before click gets interpreted as drag
# ClickMotionDistance=4 # [0-32]

# Delay before click gets interpreted as drag
# ClickMotionDelay=200 # [0-2000]
```

```
# Multiple click time
# MultiClickTime=400 # [0-5000]

# Delay before activating menu items
# MenuActivateDelay=10 # [0-5000]

# Delay before activating menu submenus
# SubmenuMenuActivateDelay=300 # [0-5000]

# Delay before tooltip window is displayed
# ToolTipDelay=1000 # [0-5000]

# Time before tooltip window is hidden
# ToolTipTime=1000 # [0-60000]
ToolTipTime=0

# Delay before task bar is automatically hidden
# AutoHideDelay=300 # [0-5000]

# Delay before windows are auto raised
# AutoRaiseDelay=400 # [0-5000]

# Resistance in pixels when trying to move windows off the screen (10000 =
infinite)
# EdgeResistance=32 # [0-10000]

# Delay for pointer focus switching
# PointerFocusDelay=200 # [0-1000]

# Distance in pixels before windows snap together
# SnapDistance=8 # [0-64]

# Screen edge workspace switching delay
# EdgeSwitchDelay=600 # [0-5000]

# Inital scroll bar autoscroll delay
# ScrollBarStartDelay=500 # [0-5000]

# Scroll bar autoscroll delay
# ScrollBarDelay=30 # [0-5000]

# Auto scroll start delay
# AutoScrollStartDelay=500 # [0-5000]

# Auto scroll delay
# AutoScrollDelay=60 # [0-5000]

# Bitmask of root window button click to use in window manager
# UseRootButtons=255 # [0-255]
```

```
UseRootButtons=2 # [0-255]

# Bitmask of buttons that raise the window when pressed
# ButtonRaiseMask=1 # [0-255]

# Desktop mouse-button click to show the menu
# DesktopWinMenuButton=1 # [0-20]

# Desktop mouse-button click to show the window list
# DesktopWinListButton=2 # [0-5]

# Desktop mouse-button click to show the window list menu
DesktopMenuButton=2 # [0-20]

# TitleBar mouse-button double click to maximize the window
# TitleBarMaximizeButton=1 # [0-5]

# TitleBar mouse-button double clock to rollup the window
# TitleBarRollupButton=2 # [0-5]

# Delay between new-mail checks. (seconds)
# MailCheckDelay=30 # [0-86400]

# Width of CPU Monitor
# TaskBarCPUSamples=20 # [2-1000]


# Titlebar buttons from left to right (x=close, m=max, i=min, h=hide, r=rollup,
s=sysmenu, d=depth)
# TitleButtonsLeft="s"

# Titlebar buttons from right to left (x=close, m=max, i=min, h=hide, r=rollup,
s=sysmenu, d=depth)
# TitleButtonsRight="xmir"

# Titlebar buttons supported by theme (x,m,i,r,h,s,d)
# TitleButtonsSupported="xmis"

# Icon search path (colon separated)
# IconPath=""

# Mailbox path (use $MAIL instead)
# MailBoxPath=""

# Command to run on mailbox
MailCommand="Eterm -t mutt"

# Command to run when new mail arrives
# NewMailCommand=""
```

```
# Command to lock display/screensaver
# LockCommand="xlock"
LockCommand="xscreensaver-command -lock"

# Command to run on clock
ClockCommand="xdaliclock -24 -font fixed -geometry -0+0 -datemode
YYMMDD"

# Command to select and run a program
# RunCommand=""
#RunCommand="grun"
#or "gnome-run"

# Displays automagic menu containing all files and opens them with this
# OpenCommand=""
#OpenCommand="gless"

# Terminal emulator must accept -e option.
TerminalCommand=Eterm

# Command to start logout
# LogoutCommand=""
#LogoutCommand="killall panel"

# Command to cancel logout
# LogoutCancelCommand=""

# Command to shutdown the system
ShutdownCommand="sudo /sbin/shutdown -h now"

# Command to reboot the system
RebootCommand="sudo /sbin/shutdown -r now"

# Command to run on CPU status
CPUStatusCommand="gtop"

# Command to run on Net status
NetStatusCommand="Eterm -g 35x1 -q --no-cursor -n NetSpeed -e netspeed"

# Command to run for address bar entries
# AddressBarCommand=""

# Network device to show status for
NetworkStatusDevice="eth0"

# Clock Time format (strftime format string)
# TimeFormat="%H:%M:%S"

# Clock Date format for tooltip (strftime format string)
# DateFormat="%B %A %Y-%m-%d %H:%M:%S %Z"
```

```
DateFormat="%A %Y-%m-%d %H:%M:%S %Z"

# Theme
##Theme="metal2/default.theme"
Theme="blueHeart/mg.theme"

# Theme Author
# ThemeAuthor=""

# Theme Description
# ThemeDescription=""

# TitleFontName="-adobe-helvetica-bold-r-*-*-*-120-*-*-*-*-*-*"
# MenuFontName="-adobe-helvetica-bold-r-*-*-*-120-*-*-*-*-*-*"
# StatusFontName="-adobe-courier-bold-r-*-*-*-120-*-*-*-*-*-*"
# QuickSwitchFontName="-adobe-courier-bold-r-*-*-*-120-*-*-*-*-*-*"
# NormalButtonFontName="-adobe-helvetica-medium-r-*-*-*-120-*-*-*-*-*-*"
# ActiveButtonFontName="-adobe-helvetica-bold-r-*-*-*-120-*-*-*-*-*-*"
# NormalTaskBarFontName="-adobe-helvetica-medium-r-*-*-*-120-*-*-*-*-*-*"
# ActiveTaskBarFontName="-adobe-helvetica-bold-r-*-*-*-120-*-*-*-*-*-*"
# MinimizedWindowFontName="-adobe-helvetica-medium-r-*-*-*-120-*-*-*-*-*-*"
# ListBoxFontName="-adobe-helvetica-medium-r-*-*-*-120-*-*-*-*-*-*"
# ToolTipFontName="-adobe-helvetica-medium-r-*-*-*-120-*-*-*-*-*-*"
# ClockFontName="-adobe-courier-medium-r-*-*-*-140-*-*-*-*-*-*"
# ApmFontName="-adobe-courier-medium-r-*-*-*-140-*-*-*-*-*-*"
# LabelFontName="-adobe-helvetica-medium-r-*-*-*-140-*-*-*-*-*-*"
# ColorDialog="rgb:C0/C0/C0"
# ColorActiveBorder="rgb:C0/C0/C0"
# ColorNormalBorder="rgb:C0/C0/C0"
# ColorNormalTitleButton="rgb:C0/C0/C0"
# ColorNormalTitleButtonText="rgb:00/00/00"
# ColorNormalButton="rgb:C0/C0/C0"
# ColorNormalButtonText="rgb:00/00/00"
# ColorActiveButton="rgb:E0/E0/E0"
# ColorActiveButtonText="rgb:00/00/00"
# ColorActiveTitleBar="rgb:00/00/A0"
# ColorNormalTitleBar="rgb:80/80/80"
# ColorActiveTitleBarText="rgb:FF/FF/FF"
# ColorNormalTitleBarText="rgb:00/00/00"
# ColorNormalMinimizedWindow="rgb:C0/C0/C0"
# ColorNormalMinimizedWindowText="rgb:00/00/00"
# ColorActiveMinimizedWindow="rgb:E0/E0/E0"
# ColorActiveMinimizedWindowText="rgb:00/00/00"
# ColorNormalMenu="rgb:C0/C0/C0"
# ColorActiveMenuItem="rgb:A0/A0/A0"
# ColorActiveMenuItemText="rgb:00/00/00"
# ColorNormalMenuItemText="rgb:00/00/00"
# ColorDisabledMenuItemText="rgb:80/80/80"
# ColorMoveSizeStatus="rgb:C0/C0/C0"
# ColorMoveSizeStatusText="rgb:00/00/00"
```

```
# ColorQuickSwitch="rgb:C0/C0/C0"
# ColorQuickSwitchText="rgb:00/00/00"
# ColorDefaultTaskBar="rgb:C0/C0/C0"
# ColorNormalTaskBarApp="rgb:C0/C0/C0"
# ColorNormalTaskBarAppText="rgb:00/00/00"
# ColorActiveTaskBarApp="rgb:E0/E0/E0"
# ColorActiveTaskBarAppText="rgb:00/00/00"
# ColorMinimizedTaskBarApp="rgb:A0/A0/A0"
# ColorMinimizedTaskBarAppText="rgb:00/00/00"
# Color for windows on other workspaces
# ColorInvisibleTaskBarApp="rgb:80/80/80"

# ColorInvisibleTaskBarAppText="rgb:00/00/00"
# ColorScrollBar="rgb:A0/A0/A0"
# ColorScrollBarArrow="rgb:C0/C0/C0"
# ColorScrollBarSlider="rgb:C0/C0/C0"
# ColorListBox="rgb:C0/C0/C0"
# ColorListBoxText="rgb:00/00/00"
# ColorListBoxSelection="rgb:80/80/80"
# ColorListBoxSelectionText="rgb:00/00/00"
# ColorToolTip="rgb:E0/E0/00"
# ColorToolTipText="rgb:00/00/00"
# ColorClock="rgb:00/00/00"
# ColorClockText="rgb:00/FF/00"
# ColorApm="rgb:00/00/00"
# ColorApmText="rgb:00/FF/00"
# ColorLabel="rgb:C0/C0/C0"
# ColorLabelText="rgb:00/00/00"
# ColorInput="rgb:FF/FF/FF"
# ColorInputText="rgb:00/00/00"
##ColorInputSelection="rgb:00/00/80"
##ColorInputSelectionText="rgb:FF/FF/FF"
#DesktopBackgroundColor="rgb:00/50/60"
##DesktopBackgroundColor=""
##DesktopBackgroundImage="/mnt/c/media/WALLPAPERS/blackmarble.jpg"
# ColorCPUStatusUser="rgb:00/FF/00"
# ColorCPUStatusSystem="rgb:FF/00/00"
# ColorCPUStatusNice="rgb:00/00/FF"
# ColorCPUStatusIdle="rgb:00/00/00"
# ColorNetSend="rgb:FF/FF/00"
# ColorNetReceive="rgb:FF/00/FF"
# ColorNetIdle="rgb:00/00/00"

# KeyWinRaise="Alt+F1"
# KeyWinOccupyAll="Alt+F2"
# KeyWinLower="Alt+F3"
# KeyWinClose="Alt+F4"
# KeyWinRestore="Alt+F5"
# KeyWinPrev="Alt+Shift+F6"
# KeyWinNext="Alt+F6"
```

```
# KeyWinMove="Alt+F7"
# KeyWinSize="Alt+F8"
# KeyWinMinimize="Alt+F9"
# KeyWinMaximize="Alt+F10"
# KeyWinMaximizeVert="Alt+Shift+F10"
# KeyWinHide="Alt+F11"
# KeyWinRollup="Alt+F12"
# KeyWinMenu="Alt+Space"
# KeySysSwitchNext="Alt+Tab"
# KeySysSwitchLast="Alt+Shift+Tab"
# KeySysWinNext="Alt+Esc"
# KeySysWinPrev="Alt+Shift+Esc"
# KeySysWinMenu="Shift+Esc"
# KeySysDialog="Alt+Ctrl+Del"
# KeySysMenu="Ctrl+Esc"
# KeySysRun="Alt+Ctrl+r"
# KeySysWindowList="Alt+Ctrl+Esc"
# KeySysAddressBar="Alt+Ctrl+Space"
# KeySysWorkspacePrev="Alt+Ctrl+Left"
# KeySysWorkspaceNext="Alt+Ctrl+Right"
# KeySysWorkspacePrevTakeWin="Alt+Ctrl+Shift+Left"
# KeySysWorkspaceNextTakeWin="Alt+Ctrl+Shift+Right"
# KeySysWorkspace1="Alt+Ctrl+1"
# KeySysWorkspace2="Alt+Ctrl+2"
# KeySysWorkspace3="Alt+Ctrl+3"
# KeySysWorkspace4="Alt+Ctrl+4"
# KeySysWorkspace5="Alt+Ctrl+5"
# KeySysWorkspace6="Alt+Ctrl+6"
# KeySysWorkspace7="Alt+Ctrl+7"
# KeySysWorkspace8="Alt+Ctrl+8"
# KeySysWorkspace9="Alt+Ctrl+9"
# KeySysWorkspace10="Alt+Ctrl+0"
# KeySysWorkspace11="Alt+Ctrl+["
# KeySysWorkspace12="Alt+Ctrl+]"
# KeySysWorkspace1TakeWin="Alt+Ctrl+Shift+1"
# KeySysWorkspace2TakeWin="Alt+Ctrl+Shift+2"
# KeySysWorkspace3TakeWin="Alt+Ctrl+Shift+3"
# KeySysWorkspace4TakeWin="Alt+Ctrl+Shift+4"
# KeySysWorkspace5TakeWin="Alt+Ctrl+Shift+5"
# KeySysWorkspace6TakeWin="Alt+Ctrl+Shift+6"
# KeySysWorkspace7TakeWin="Alt+Ctrl+Shift+7"
# KeySysWorkspace8TakeWin="Alt+Ctrl+Shift+8"
# KeySysWorkspace9TakeWin="Alt+Ctrl+Shift+9"
# KeySysWorkspace10TakeWin="Alt+Ctrl+Shift+0"
# KeySysWorkspace11TakeWin="Alt+Ctrl+Shift+["
# KeySysWorkspace12TakeWin="Alt+Ctrl+Shift+]"

WorkspaceNames="  1  ",  "  2  ",  "  3  ",  "  4  "
```